Alessandro Vivas Andrade
Leonardo Carneiro de Araújo
Cristiano Grijó Pitangui
Luciana Pereira de Assis

LINUX

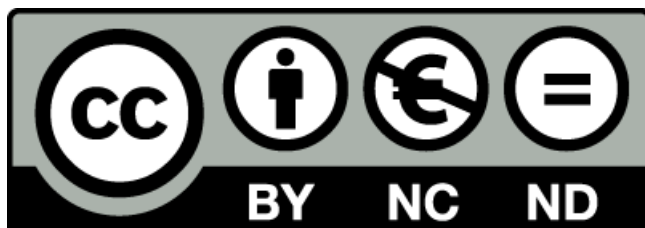
comandos básicos e avançados

2ª edição

2019

Alessandro Vivas Andrade, Leonardo Carneiro Araújo,
Cristiano Grijó Pitangui, Luciana Pereira de Assis

LINUX: COMANDOS BÁSICOS E AVANÇADOS

Diamantina
2019

EDITOR: Alessandro Vivas Andrade
PROJETO GRÁFICO: Alessandro Vivas Andrade
CAPA: Gabriela Miranda de Souza

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

L761 Linux [recurso eletrônico] : comandos básicos e avançados / Alessandro Vivas Andrade... [et al.]. – 2.ed. – Diamantina, MG: Ed. do Autor, 2019.

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-85-920329-2-0

1. Linux (Sistema operacional de computador). 2. Sistemas operacionais (Computadores). I. Andrade, Alessandro Vivas. II.Araújo, Leonardo Carneiro. III. Pitangui, Cristiano Grijó. IV. Assis, Luciana Pereira de.

CDD 005.43

Elaborado por Maurício Amormino Júnior – CRB6/2422

Prefácio da Segunda Edição

A primeira edição do livro Linux: comandos Básicos e Avançados foi publicada em novembro de 2015. Organizada em 15 capítulos e 141 páginas atingiu milhares de downloads e recebemos diversas sugestões de leitores para ampliação da obra.

A segunda edição está organizada em 19 capítulos divididos em 267 páginas. Os capítulos da primeira edição foram totalmente revisados e novos conteúdos foram adicionados.

O intuito da obra é agregar em um único livro um manual didático e prático do uso de comandos do sistema operacional Linux para as áreas de computação e engenharia.

Este livro pode ser utilizado como bibliografia principal ou complementar em cursos de Ciência da Computação ou Sistemas de Informação para as disciplinas de Sistemas Operacionais e Redes de Computadores.

Encontrou algum erro no livro ou tem alguma sugestão? Favor encaminhar e-mail para
alessandro.vivas@gmail.com.

Prefácio da Primeira Edição

Este livro tem como objetivo apresentar de uma maneira simples e didática os principais comandos do **shell** do Sistema Operacional Linux. Nele são abordados a maioria dos comandos disponíveis pelo Sistema Operacional Linux onde grande parte destes são compatíveis com o Sistema Operacional MacOs. Reunimos profissionais com formações experiências e formações distintas para apresentar visões diferentes deste mesmo tema.

A motivação de escrever este livro surgiu a partir de três realidades distintas, a primeira de reunir em um único material um conteúdo para auxiliar alunos dos cursos de Computação e Engenharia que trabalham com o Sistema Operacional Linux. A segunda surgiu de apresentar conceitos úteis para os administradores de sistemas Linux. A terceira motivação foi de reunir em um único material comandos úteis para os pesquisadores de todas as áreas que trabalham com o Linux.

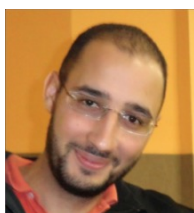
Este livro pode ser utilizado como bibliografia principal ou complementar em cursos de Ciência da Computação ou Sistemas de Informação para as disciplinas de Sistemas Operacionais e Redes de Computadores.

Encontrou algum erro no livro ou tem alguma sugestão? Favor encaminhar e-mail para
alessandro.vivas@gmail.com.

Sobre os Autores

Alessandro Vivas Andrade é natural de Lavras/MG e atualmente reside em Diamantina/MG. É professor do Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM) onde leciona as disciplinas de Redes de Computadores e Sistemas Distribuídos. Também atua como Professor do Curso de Mestrado em Programa de Pós-Graduação em Educação em Ciências, Matemática e Tecnologia (PPGECMAT/UFVJM). Graduou-se em Engenharia Elétrica (UFMG) e depois cursou Mestrado e Doutorado ambos em Engenharia Elétrica na UFMG. Tem interesses nas áreas de Inteligência Artificial, Aprendizado de Máquina e Ciência de Dados.

Luciana Pereira de Assis é natural de Belo Horizonte/MG e atualmente reside em Diamantina/MG. É professora do Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM) onde leciona as disciplinas de Algoritmos e Estrutura de Dados, Pesquisa Operacional e Inteligência Artificial. Também atua como Professor do Curso de Mestrado em Gestão em Instituições de Ensino (PPGGIED/UFVJM). Graduou-se em Ciência da Computação (UNIBH) e depois cursou Mestrado em Ciência da Computação (UFMG) e Doutorado em Engenharia Elétrica na UFMG. Tem interesses nas áreas de Otimização, Inteligência Artificial e Análise de Redes Sociais.

Leonardo Carneiro de Araújo é natural de Belo Horizonte/MG. É professor da Universidade Federal de São João del Rei (UFSJ) onde leciona as disciplinas de Teoria da Informação, Processamento de Áudio e Vídeo, Análise de Sinais e Sistemas, dentre outras. Graduou-se em Engenharia Elétrica (UFMG), depois cursou Mestrado e Doutorado em Engenharia Elétrica (UFMG). Tem interesse nas áreas de Teoria da Informação, Linguística e Linguística Quantitativa, Reconhecimento de Fala, Reconhecimento de Padrões e Inteligência Artificial.

Cristiano Grijó Pitangui é natural de Ouro Branco/MG e atualmente reside em Ouro Branco/MG. É professor do Curso de Sistemas de Informação da Universidade Federal de São João del-Rei onde leciona as disciplinas na área de Computação. Também atua como Professor do Curso de Mestrado em Gestão em Instituições de Ensino (PPGGIED/UFVJM). Graduou-se em Ciência da Computação (UFJF) e depois cursou Mestrado e Doutorado em Engenharia de Sistemas e Computação com ênfase em Inteligência Artificial na COPPE-UFRJ. Tem interesses nas áreas de Aprendizado de Máquina, Inteligência Artificial e Lógica de Primeira Ordem.

Conteúdo

Lista de Figuras

Listagens

Capítulo 1

Conceitos Básicos

Sumário

1.1 Introdução

Linux pode ser definido como um Sistema Operacional como o OSX da Apple e o Windows. Um Sistema Operacional pode ser definido como um software que atua como gestor dos recursos de hardware permitindo o uso dos mesmos de forma conveniente. Quando criamos um arquivo no Linux ou em qualquer outro Sistema Operacional não precisamos entender o funcionamento detalhado do sistema de arquivos para realizar esta tarefa. Um Sistema Operacional torna o hardware mais fácil de usar criando abstrações que permitem que usuários e desenvolvedores realizem suas tarefas.

Um Sistema Operacional é o primeiro programa a ser carregado quando ligamos um computador e é responsável pelo gerenciamento do mesmo até que você o desligue.

Para usar um Sistema Operacional é necessário instalar o mesmo. Quando compramos um computador ele já vem com um Sistema Operacional instalado, mas é possível instalar outro sistema ou atualizar o existente.

Existem várias versões do Sistema Operacional Linux e cada uma tem uma finalidade bem definida. Alguns segurança, outros usabilidade, outros para o ambiente educacional, etc... O primeiro passo é escolher uma distribuição de Linux para realizar o procedimento de instalação.

Podemos definir uma Distribuição Linux como um kernel (Sistema Operacional) e mais um conjunto de aplicativos. Assim, as distribuições diferem na maneira que organizam estes aplicativos bem como no uso de diferentes gerenciadores de interface gráfica.

O shell ou interpretador de comandos não é parte do Sistema Operacional do ponto de vista de arquitetura. Apesar de não fazer parte, ele utiliza massivamente os recursos do Sistema Operacional [9, 7].

1.2 Instalação do Sistema Operacional Linux

Existem basicamente três maneiras de instalar e acessar o Linux: instalação como sistema único, instalação em alguma partição em conjunto com outro sistema operacional, ou utilizar um software de virtualização e executar o Linux dentro de outro Sistema Operacional.

As Distribuições de Linux disponibilizam um arquivo compactado para que usuários possam fazer o download e realizar o procedimento de instalação. Este arquivo compactado é chamado de imagem.

A imagem da Distribuição Ubuntu pode ser obtida em `https://ubuntu.com/download/desktop`, o Debian pode ser obtido em `https://www.debian.org/distrib/index.pt.html`, Slackware em `http://www.slackware.com/getslack/`, o Fedora em `https://getfedora.org/pt_BR/workstation/download/` e o OpenSuse em `https://software.opensuse.org/` . Escolha uma das distribuições e baixe o arquivo contendo a imagem do Sistema Operacional. Após o down-

load, é necessário gravar este arquivo em um formato específico para ser possível realizar o procedimento de instalação.

O software UNetbootin `https://unetbootin.github.io/` é uma excelente opção para transformar seu arquivo (imagem) em uma versão de instalação da distribuição. Para isso você apenas necessitará de um pendrive e o arquivo de instalação.

A Figura 1.1 apresenta a tela do software UNetbootin. Você apenas precisa selecionar o arquivo de instalação e a porta USB onde está o pendrive onde será instalado o arquivo.

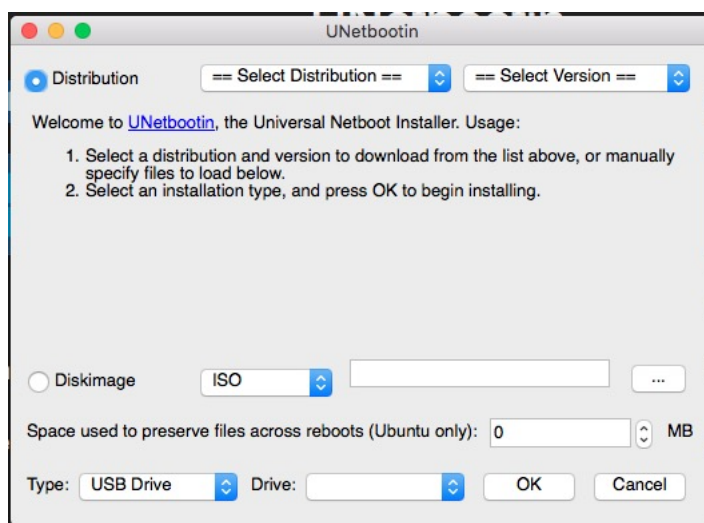



Figura 1.1: UNetbootin

Você pode usar todo o seu disco rígido ou instalar em uma partição do seu disco. Neste caso teremos um gerenciador de boot e no momento da inicialização da máquina você deverá escolher o sistema operacional desejado.

A terceira forma é executar o sistema operacional Linux ao mesmo tempo que outro sistema operacional. Para isto é necessário instalar um programa de virtualização, como por exemplo **Virtualbox** [10] ou outro similar. Este programa permite rodar dois sistemas operacionais ao mesmo tempo. Lembre-se que neste caso vamos precisar de mais memória RAM para conseguirmos um desempenho razoável.

Não é objetivo deste livro explicar os processos de instalação, pois são diferentes para cada distribuição. Caso tenha alguma dúvida consulte o manual de instalação da distribuição escolhida.

1.3 Qual Distribuição?

Esta é uma pergunta difícil de responder, pois isto depende do gosto do usuário. Comece sempre de uma distribuição que o procedimento de instala-

ção seja bem simplificado. Por esta razão sempre indicamos as distribuições **Ubuntu** [1], **Fedora** [3], **Debian** [2] ou **OpenSuse** [5].

1.4 Acessando o Sistema Operacional Linux

Existem várias maneiras de acessar o Linux e isto vai depender da forma que instalou o sistema. A maneira mais simples é entrar diretamente após o **boot** do computador quando apenas um sistema existe no disco rígido.

Se você compartilhou o disco rígido com outros sistemas operacionais, terá que realizar a seleção no momento do boot. Assim, ao ligar o computador, será apresentado o gerenciador de boot **grub** ou **lilo** . Caso a máquina já esteja ligada e exibindo a janela de login do Windows, deve-se reinicializá-la, pressionando a combinação de teclas <CTRL+ALT+DEL>. Escolha a opção Linux e aperte a tecla *ENTER*.

1.5 Acessando o Terminal do Linux

Existem duas maneiras de acessar o terminal (shell) do Linux. A primeira maneira é entrar na parte gráfica e selecionar o shell. A segunda maneira é utilizar os terminais virtuais presentes na distribuição.

Para acessar os terminais virtuais basta pressionar <CTRL+ALT+F1> para obter o primeiro dos 6 terminais (F1 a F6) virtuais disponíveis para se acessar o sistema.

O Linux possibilita o uso de até 63 terminais virtuais simultaneamente. Por default, são disponibilizados os 6 terminais, onde o usuário pode executar tarefas distintas em cada um deles.

Atalhos para os terminais:

- <CTRL+ALT+FN>: vai da interface gráfica para a console N (onde N vale de 1 a 6);
- <CTRL+ALT+F1>: vai para console 1;
- <CTRL+ALT+F2>: vai para console 2;
- <CTRL+ALT+F3>: vai para console 3;
- ...
- <CTRL+ALT+F6>: vai para console 6;
- <CTRL+ALT+F7>: volta para a interface gráfica.

1.6 Entrando no Sistema

Escolha um terminal modo texto e entre no sistema. A Listagem 1.1 ilustra o procedimento de acesso. A palavra **Login** indica o nome do usuário e **Password** indica a senha do usuário.

Listagem 1.1: Acessando o Sistema

```
Login: <digite o seu login>
Password: <digite a sua senha>
```

Para logar no sistema você deve ser previamente cadastrado no sistema pelo administrador. Ao digitar a senha, o Linux consultará o arquivo **passwd** , localizado no diretório /**etc**. Caso as informações estejam corretas, o sistema permitirá o acesso. Durante o processo de instalação ocorre este cadastramento de usuário.

Ao realizar este processo, ocorre a distinção entre usuários, sendo permitido que várias pessoas possam usar a mesma máquina simultaneamente e que somente você (usuário) tenha acesso aos seus arquivos. Você não tem permissão para apagar ou modificar arquivos do sistema; isto é a grande diferença do Linux. Apenas o administrador tem o privilégio de acesso a estes arquivos.

No Linux chamamos o administrador do sistema de **root** e somente ele tem privilégios para fazer alterações no sistema operacional. Por isto não fique com medo de danificar alguma coisa no sistema, pois isto só poderá ser realizado se tiver privilégio do usuário **root**.

1.7 Significado do Shell

Se você teve sucesso no processo de login, irá aparecer um prompt. O símbolo do prompt não é fixo e varia conforme o interpretador de comandos (**shell**) ou configuração, Listagem 1.2.

Listagem 1.2: Prompt de Login

```
Last login: Fri Sep  4 14:13:10 2015 from
   192.168.0.101
[avivas@musashi ~]$
```

Onde *avivas* é o nome do usuário e **musashi** é o nome da máquina. Deseja alterar o nome de sua máquina? Realize o procedimento descrito na Listagem 1.3.

Listagem 1.3: Alterando o Nome da Máquina

```
# logar como root
$ su -
```

```
Senha:
# edite o arquivo hostname
[root@musashi ~]# vi /etc/hostname
musashi.vivascorp
# salve o arquivo
```

Para os usuários comuns o prompt é o sinal $ e pode também conter o nome do diretório em que você está naquele instante (diretório corrente). Geralmente, ao entrar no sistema, um interpretador de comandos - shell - é iniciado, o qual está associado à sua conta em seu diretório home. A Listagem 1.4 apresenta o formato apresentado no terminal.

Listagem 1.4: Explicando a Padronização de Apresentação

```
/home/jose$
```

Diretórios no Linux/UNIX são especificados por uma / e não uma \, diferentemente do que é definido para outros sistemas, como por exemplo o DOS.

1.8 Formato dos comandos

Os comandos (arquivos executáveis, chamados também de programas) no Linux, passados via shell, possuem a seguinte forma:

- <comando> → ls
- <comando><espaço><opções> → ls -lF
- <comando><espaço><opções><espaço><argumentos> → cp -R /home/vivas/teste /home/vivas/ensino/

A quase totalidade dos comandos possui todos os três elementos acima. A Listagem 1.5 apresenta o comando de copiar um arquivo de um diretório para outro arquivo que está em outro diretório, preservando os atributos do arquivo (permissões, dono, marca de tempo).

Listagem 1.5: Formato dos Comandos

```
cp -p /home/origem/texto.txt /home/destino/texto.
    txt
```

Entretanto, existem alguns comandos que possuem apenas opções, apenas argumentos ou nenhum destes, i.e., somente o próprio comando é necessário. O comando **clear**, utilizado para limpar a tela do terminal, não possui argumento algum.

É importante atentar para a existência ou não de espaços entre os caracteres ao se definir uma ação completa (comando <espaço> opção1 <espaço>

opção2 <espaço> opção3 <espaço> argumento1 <espaço> argumento2 <espaço> argumento3).

Existe uma flexibilidade para se passar opções ao sistema operacional. Quando há a necessidade de se especificar mais de uma opção, o usuário pode utilizar um aninhamento de opções, i.e., usar o mesmo hífen para especificar mais de uma opção:

- <comando> -<opção1> -<opção2> -<opção3>
- <comando> -<opção1opção2opção3>

Toda opção é precedida de um ou dois hífens(-), colocado sem espaços. Os dois hifens são usados para opções por extenso. Ex:

Listagem 1.6: Opções dos Comandos

```
$ls<espaco>-a<espaco>-l
$ls<espaco>--all
$pwd<espaco>--version
$cd<espaco>--help
```

A opção é definida alternativamente por uma letra (-a, -l,-H) ou por extenso(–color,–size,–count). O uso de dois hifens isolados na linha de comando estabelece para o interpretador que não haverá mais opções a serem passadas para se efetuar aquele comando. Caso exista algo do tipo -texto escrito logo em seguida aos dois hifens em sequência, este não será interpretado pelo shell como uma opção, mas sim como um argumento. A Listagem 1.7 apresenta um exemplo, no qual a opção -F não será interpretada como opção, conforme explicado anteriormente.

Listagem 1.7: Opções Múltiplas

```
vivas@musashi:/usr/local$ grep -- -a /etc/hosts
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

1.9 Shells

Ao entrar no Linux, o sistema habilita um shell para trabalho. O **shell** (interpretador de comandos) default, especificado pela configuração inicial, é o **bash** (localizado no diretório */bin/bash*), mas pode-se alterar essa escolha.

- bash - $ - Bourne Again Shell. O shell mais utilizado (e mais poderoso) do Linux. Criado e distribuído pelo projeto GNU. Oferece comandos de edição de linha, substituição baseado no histórico e compatibilidade com o Bourne shell(sh).

- csh - % C shell. Desenvolvido em Berkeley. Compatível com Bourne Shell para uso interativo, mas tem uma interface diferente de programação. Não oferece comandos de edição.
- ksh - Korn shell - O shell mais popular do Unix e o primeiro a introduzir as técnicas modernas de shell no Bourne shell. Oferece comandos de edição de linha.
- sh - $ - Bourne Shell. Shell original do linux. Não oferece comandos de edição.
- zsh - z shell. O mais novo dos shells. Compatível com Bourne shell e oferece comandos de edição.
- tcsh - % - um C shell melhorado.

1.9.1 Descobrindo o Shell

Para saber qual shell você está utilizando basta digitar o comando apresentado na Listagem 1.8. O comando **echo** é utilizado para imprimir variáveis de ambiente ou textos no terminal.

Listagem 1.8: Shell Utilizado

```
[avivas@musashi ~]$ echo $SHELL
/bin/bash
```

Para saber quais interpretadores de comandos estão instalados em seu sistema utilize o comando apresentado na Listagem 1.9. O comando **cat** é utilizado para concatenar arquivos, ou entrada padrão, e imprimir o resultado na saída padrão, sendo, muitas vezes utilizado para ler conteúdos de arquivos, visualizando-os no terminal.

Listagem 1.9: Shells Instalados

```
[avivas@musashi ~]$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
```

1.10 Case Sensitive

O sistema é do tipo Case Sensitive, i.e., letras maiúsculas são diferenciadas de minúsculas. Sendo assim, os arquivos Exemplo.txt, Exemplo.TXT, ExEmPlO.tXt e EXEMPLO.txt são entidades diferentes para o sistema. Isso inclui

comandos, programas, opções de comandos e argumentos. Como iremos ver, tudo no Linux é tratado como arquivo.

1.11 Movimentação no terminal

- Apagar um caractere à esquerda: <backspace>
- Apagar uma linha inteira: <CTRL+U>
- Andar na linha de comandos: para percorrer os caracteres na linha do shell basta utilizar a seta de direção para esquerda (o cursor move para o próximo caractere à esquerda) ou seta para direita (o cursor move para o próximo caractere à direita)
- Apagando o caractere localizado sobre o cursor: <delete>
- Mover o cursor para o início da linha de comandos: <CTRL+A>
- Mover o cursor para o fim da linha de comandos: <CTRL+E>
- Apagar todos os caracteres localizados à esquerda do cursor: <CTRL+U>
- Copiar um conteúdo: o conteúdo recentemente apagado é copiado com a combinação <CTRL+Y>
- Apagar o que estiver à direita do cursor: <CTRL+K>

A combinação <CTRL+D> ao ser usada numa linha que contenha um grupo de caracteres, desempenhará a função da tecla <delete>. Caso não exista nada na linha de comando corrente, essa combinação desempenhará a função de logout.

1.12 Primeiros comandos

Um comando é um software que realiza uma determinada função - usualmente uma função especializada. Nos sistemas Unix, comando é um simples arquivo localizado geralmente no diretório */bin* ou */sbin*. Assim, define-se como caminho absoluto aquele caminho completo, desde o diretório raiz. A Listagem 1.10 apresenta o caminho completo para o comando **ls** .

Listagem 1.10: Caminho Completo

```
// caminho completo
$ /bin/ls
```

A seguir são dados os comandos mais básicos. Para limpar a tela do terminal, use o comando clear ou a combinação <CTRL+L>. O cursor será posicionado no canto superior esquerdo:

Listagem 1.11: Limpando a Tela

```
$ clear
```

1.13 Visualizando textos longos no terminal

Para visualizar textos longos no terminal que, por ventura, tenham sido escondidos na parte superior da tela, basta pressionar as teclas:

- <SHIFT+PAGEUP>
- <SHIFT+PAGEDOWN>

Essas teclas rolarão o conteúdo que ficou além ou aquém do espaço de tela para baixo ou para cima.

Quando apagamos o conteúdo da tela com o comando clear(ou <CTRL+L>), na verdade estamos reposicionando o cursor. Isso quer dizer que parte da informação exibida na tela pouco antes da execução desse comando é deslocada para além do limite superior da tela. Para comprovar o fato, experimente limpar o conteúdo da tela e depois teclar <SHIFT+PAGEUP>.

1.14 Exibindo Mensagens

O comando **echo** disponibiliza mensagens na saída padrão (vídeo). Além disso, ele é usado para visualizar o conteúdo de variáveis de shell, que serão estudadas mais adiante. Exemplos na Listagem 1.12.

Listagem 1.12: Exibindo Mensagens no Terminal

```
[avivas@musashi ~]$  echo teste        ou
[avivas@musashi ~]$  echo 'teste'  ou
[avivas@musashi ~]$  echo"teste"
[avivas@musashi ~]$  echo -e "string1\tstring2\
   nstring3"
```

A opção -e habilita a interpretação de caracteres especiais, tais como:

- \ : nova linha
- \\ : barra invertida
- $\backslash t$: tabulação horizontal
- $\backslash v$: tabulação vertical
- $\backslash r$: retorno de linha
- $\backslash nnn$: código ASCII correspondente

1.15 Histórico do Terminal

Para facilitar as coisas, o Linux mantém o histórico dos comandos digitados, tanto válidos quanto inválidos. Isto evita que você fique perdendo tempo em digitar tudo novamente. Assim, para navegar entre os últimos comandos passados ao sistema, utilize as setas direcionais (↑ ou ↓). Ao apertar diversas vezes, tais comandos irão aparecer na ordem cronológica inversa, i.e., do mais recente para o mais antigo.

Quando se tem um histórico com poucos comandos a navegação por setas direcionais pode ser feita sem problemas. No entanto, quando a lista passa a contar com 50 ou mais comandos, a busca de um dado comando passa a ser enfadonha. Nesse caso, utilize o mecanismo de procura <CTRL+R>, cuja interface é apresentada na Listagem 1.13.

Listagem 1.13: Histórico

```
(reverse-i-search)‘’:
// comece a digitar aquele comando find
(reverse-i-search)‘fi’: find / | grep a
// basta digitar enter para executar o comando
```

Ao digitar o primeiro caractere, surgirá o comando mais recente que possui aquele caractere. Para refinar a seleção, deve-se continuar digitando outros caracteres e o comando mais próximo da sequencia digitada irá aparecer ao lado. Para executar a escolha reconhecida na busca basta apertar a tecla <ENTER>. Para editar o comando a tecla <backspace> deve ser usada.

Algumas vezes pode ser necessário editar algum comando do histórico antes de executá-lo. Para tanto você deverá utilizar as setas direcionais para direita ou esquerda (← ou →) quando ver o comando desejado para editá-lo antes de executar.

1.15.1 Comando *history*

O comando **history** pode ser executado para listar o histórico de comandos utilizados no Terminal. A Listagem 1.14 apresenta o resultado da execução do comando.

Listagem 1.14: Comando history

```
[avivas@musashi ~]$ history
1 tar cvf etc.tar /etc/
2 cp /etc/passwd /backup
3 ps -ef | grep http
4 service sshd restart
5 /usr/local/apache2/bin/apachectl restart
```

Os comandos armazenados no histórico são apresentados em ordem cronológica e numerados. Para executar um dos comandos anteriores, basta utilizar exclamação e o número do comando. Por exemplo, para executar novamente o quarto comando, basta fazer

Listagem 1.15: Excutar um determinado comando do history

```
[avivas@musashi ~]$  !4
service sshd restart
```

Para procurar um determinado comando no *history*, basta usá-lo em combinação com o comando *grep*. O exemplo abaixo na Listagem 1.16 ilustra o caso em que desejamos localizar um comando utilizado que contenha a palavra-chave *sshd*.

Listagem 1.16: Exemplo de busca no history

```
[avivas@musashi ~]$  history | grep sshd
4 service sshd restart
6 history | grep sshd
```

Você pode limpar todo o histórico utilizando o comando exemplificado na Listagem 1.17.

Listagem 1.17: Limpar o histórico

```
[avivas@musashi ~]$ history -c
```

O tamanho máximo do histórico é definido pela variável de ambiente *HISTSIZE* . Você poderá verificar o valor desta variável e modificá-lo, se julgar necessário, conforme exemplificado a seguir.

Para listar o valor atual da variável de ambiente que define o tamanho máximo utilize o código da Listagem 1.18.

Listagem 1.18: Tamanho do histórico

```
[avivas@musashi ~]$ echo $HISTSIZE
1000
```

Suponha que deseja aumentar o tamanho para 100 linhas, então proceda como na Listagem 1.19.

Listagem 1.19: Aumentando o Tamanho de Comandos Armazenados

```
[avivas@musashi ~]$  export HISTSIZE=100
```

Suponha que se deseja diminuir o tamanho para 0 linhas, isto é, não será armazenada mais nenhuma linha. Para isto proceda como na Listagem 1.20.

Listagem 1.20: Reduzindo o Tamanho de Comandos Armazenados

```
[avivas@musashi ~]$  export HISTSIZE=0
```

Quando o shell é inicializado, o histórico de comandos é iniciado no arquivo ~/.bash_history. O histórico de comandos pode então ser acessado através deste arquivo. Verifique digitando o comando na Listagem 1.21.

Listagem 1.21: Arquivo contendo o histórico de comandos

```
[avivas@musashi ~]$  cat ~/.bash_history
```

1.16 Usuário Root

No Linux todas as tarefas administrativas são realizadas pelo administrador do sistema também conhecido como **root**. Somente ele pode executar as operações administrativas da sua máquina, como instalação de pacotes, adicionar usuários, verificação de arquivos de log, etc...

Caso deseje executar algum comando que necessita de privilégio, é necessário acessar este usuário. A maneira de mudança de usuário depende do sistema utilizado. No sistema Debian é necessário que você realize o procedimento da Listagem 1.22. Assim, você poderá realizar todas as atividades administrativas.

Listagem 1.22: Usuário Root no Debian

```
vivas@masamune:~$ su -
Senha:
root@masamune:~#  apt install vim
```

Para o Ubuntu você precisa digitar o comando sudo antes de qualquer comando privilegiado como na Listagem 1.23.

Listagem 1.23: Usuário Root no Ubuntu

```
vivas@masamune:~$ sudo apt install vim
Password:
```

Capítulo 2

Ligando e Desligando o Linux

Sumário

2.1 Saindo do sistema

2.1.1 Saindo do Sistema com Logout

Ao terminar seu trabalho você deve sair do sistema, o comando **logout** é utilizado para fechar sua conta para que outras pessoas não entrem no seu sistema e acessem seus arquivos. A sintaxe é bastante simples e funciona quando você entrou no sistema via terminal. O procedimento é apresentado na Listagem 2.1.

Listagem 2.1: Comando logout

```
[avivas@musashi ~]$ logout
```

2.2 Saindo dom Sistema com Exit

Você também pode sair do terminal usando o comando **exit** . Apesar de serem praticamente iguais, o **exit** pode ser utilizado em qualquer script enquanto o **logout** não. O procedimento de uso do comando **exit** é apresentado Listagem 2.2.

Listagem 2.2: Comando exit

```
[avivas@musashi ~]$ exit
```

O atalho <CTRL+D> pode ser utilizado quando desejamos fechar um terminal, mas geralmente não funciona quando utilizamos interfaces gráficas.

2.3 Desligando e Reinicializando o Sistema

Outra forma de sair do sistema é desligando a máquina. Nunca desligue a máquina sem os comandos apropriados, pois isto pode corromper o sistema de arquivos do Linux. Ao se desligar a máquina corretamente, o Linux finalizará os programas, gravará os dados no disco rígido e começará a mostrar procedimentos de finalização (FS, sinais KILL, SIGTERM para os processos residentes na memória).

2.3.1 Desligando Imediatamente

Para desligar o computador utilizamos o comando **shutdown** . Caso deseje desligar imediatamente, utilize o código da Listagem 2.3. Ele vai pedir a senha do administrador para desligar a máquina. A opção **h** significa que é para parar o computador. Para desligar a máquina, você terá que ter a senha de **root**.

Listagem 2.3: Desligando Imediatamente com shutdown

```
[avivas@musashi ~]$ shutdown -h now
==== AUTHENTICATING FOR org.freedesktop.login1.power-
   off ===
É necessária autenticação para desligar o sistema.
Authenticating as: root
Password:
```

Outra maneira de desligar a máquina é utilizar o comando **poweroff** . Para utilizá-lo veja a Listagem 2.4. Ele funciona da mesma maneira que o comando **shutdown -h now**.

Listagem 2.4: Desligando Imediatamente com poweroff

```
$ sudo poweroff
```

2.3.2 Desligando após um determinado tempo

Se você quer desligar a máquina após 3 minutos, utilize o mesmo comando, mas com o argumento tempo, como na Listagem 2.5.

Listagem 2.5: Desligando Após Determinado Intervalo de Tempo

```
$ sudo shutdown -h +3 "Desligando␣"
```

Se outra pessoa estiver logada no sistema irá receber mensagens parecidas com as da Listagem 2.6.

Listagem 2.6: Mensagens recebidas

```
$
Espalhar mensagem de vivas@zafu
        (/dev/pts/0) em 9:14 ...

The system is going down for halt in 2 minutes!
Desligando

Espalhar mensagem de vivas@zafu
        (/dev/pts/0) em 9:15 ...

The system is going down for halt in 1 minute!
Desligando

Espalhar mensagem de vivas@zafu
        (/dev/pts/0) em 9:16 ...
```

```
O sistema esta sendo paralisado AGORA!
Desligando
```

2.3.3 Desligando em uma hora específica

Para desligar em uma hora determinada basta passar a hora desejada como argumento. A Listagem 2.7 apresenta o comando para desligar a máquina às 10:10 da manhã.

Listagem 2.7: Desligando Imediatamente

```
$ sudo shutdown -h 10:10 "Desligando␣"
```

2.3.4 Cancelando um shutdown

Quer interromper o comando de shutdown? Vamos supor que tenha digitado o seguinte comando da Listagem 2.8.

Listagem 2.8: Desligando em 5 minutos

```
$ sudo shutdown -h +5 "Desligando␣em␣5␣minutos"
```

Para cancelar um **shutdown** vá em outro terminal e digite o comando da Listagem 2.9.

Listagem 2.9: Cancelando Shutdown

```
$ sudo shutdown -c
shutdown: Desligamento cancelado
```

Outra maneira de cancelar o desligamento da máquina é ir no terminal e digitar <CONTROL+C>.

2.4 Reinicializado a máquina

Para reinicializar uma máquina, podemos utilizar o comando **reboot** , Listagem 2.10.

Listagem 2.10: Reinicializando com reboot

```
$ sudo reboot
Password:
```

O comando apresentado na Listagem 2.11 tem o mesmo resultado do comando **reboot**.

Listagem 2.11: Reinicializando com shutdown

```
$ sudo shutdown -r now
Password:
```

2.4.1 Reinicializando após determinado tempo

Para programar o tempo na qual a máquina irá reinicializar utilizamos a opção **-r +tempo**. A Listagem 2.12 apresenta o comando.

Listagem 2.12: Reinicializando a Máquina após Determinado Intervalo de Tempo

```
$ sudo shutdown -r +3 "Reboot em 3 minutos"
Password:
```

2.4.2 Reinicializando em uma determinada hora

Para programar a hora na qual a máquina irá reinicializar utilizamos a opção **-r +tempo**. A Listagem 2.13 apresenta o comando.

Listagem 2.13: Reinicializando a Máquina em Horário Específico

```
$ sudo shutdown -r +8:25  "Reboot as 08:25h"
Password:
```

Capítulo 3

Operações em Diretórios e Arquivos

Sumário

3.1 Árvore de Diretórios

O Sistema Operacional utiliza um sistema de arquivos para permitir o armazenamento e localização de dados de forma fácil para o usuário [8]. O Sistema Operacional Linux utiliza duas estruturas para organização: diretórios e arquivos.

A estrutura de diretórios do Linux é uma árvore invertida, isto é, a raiz da árvore de diretórios é o topo. O diretório raiz é representado por uma barra (/) e é chamado de **root** ou raiz. A Figura 3.1 apresenta a árvore de diretórios do Linux.

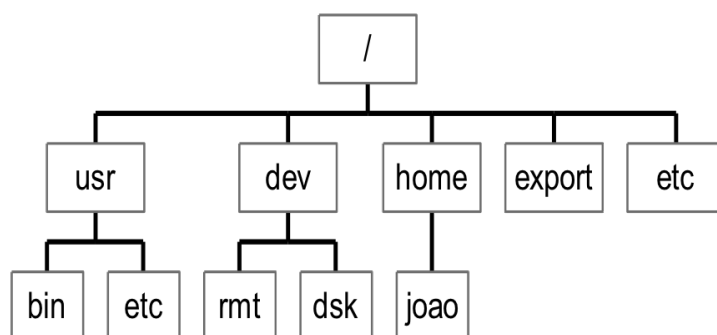



Figura 3.1: Árvore de Diretórios do Linux

3.2 Estrutura de Diretórios do Sistema Linux

A estrutura do sistema de arquivos adotada pelo sistema Linux é ilustrada na Figura 3.2. O diretório raiz (*root*) é o diretório raiz do sistema de arquivos. Apenas o usuário administrador (*root*) possui privilégio para escrever neste diretório. Note que o diretório */root* é o diretório *home* do administrador (usuário *root*), o que é diferente do diretório / (diretório raiz).

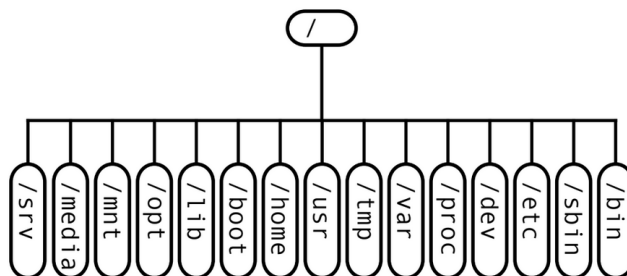



Figura 3.2: Estrutura de Diretórios do Linux

/bin contém arquivos binários executáveis, inclusive os comandos comuns utilizados, como por exemplo **ps**, **ls**, **ping**, **cp**, etc.

/sbin contém arquivos binários executáveis do sistema, como por exemplo **iptables**, **reboot**, **fdisk**, **ifconfig**, etc.

/etc contém arquivos de configuração necessários por todos os programas; contém também *scripts* de inicialização e finalização de programas

/dev contém arquivos de dispositivos como disco rígido, dispositivos usb, etc.

/proc contém informações sobre os processos do sistema

/var contém arquivos variáveis (mudam com o tempo), como por exemplo, arquivos de *log* no diretório */var/log*, arquivos de pacotes e bases de dados em */var/lib*, fila de impressão em */var/spool*; etc

/tmp contém arquivos temporários criados pelo sistema e usuários

/usr contém arquivos binários, bibliotecas, documentação e códigos fonte de programas de segundo nível

/home contém os diretórios de todos os usuários e seus arquivos pessoais

/lib contém arquivos de biblioteca que suportam os binários localizados em */bin* e */sbin*

/opt contém aplicativos de fornecedores individuais

/mnt diretório temporário de montagem de sistema de arquivos

/media diretório temporário de montagem de dispositivos removíveis, por exemplo, */media/cdrom*, */media/floppy*

/srv contém dados específicos de alguns serviços

3.3 Listando o Conteúdo do diretório

O comando **ls** executa essa tarefa. O comando **dir** (herdado de outros S.O.s) pode também existir devido a uma pré-configuração. As principais opções utilizadas são mostradas a seguir. O comando **ls**, Listagem 3.1, sem opções exibe o conteúdo na forma de uma lista.

Listagem 3.1: Listando o Conteúdo de um Diretório

```
$ ls
arquivo1.txt  arquivo2.txt
```

3.4 Listando uma única entrada por linha

O comando **ls -1** lista o conteúdo do diretório adicionando uma entrada (diretório e/ou arquivos) por linha. A Listagem 3.2 apresenta o resultado do comando.

Listagem 3.2: Uma Entrada por Linha

```
$ ls -1
afpovertcp.cfg
aliases
aliases.db
apache2
asl
asl.conf
authorization.deprecated
...
```

3.5 Listando o Conteúdo no Formato Longo

Este comando imprime informações adicionais como permissões, dono, data de criação, data de modificação, etc. A Listagem 3.3 apresenta um exemplo de uso deste comando.

Listagem 3.3: Comando ls no formato longo

```
vivas@musashi:/etc$ ls -l
total 1084
drwxr-xr-x  3 root root       4096 Apr 26  2018 acpi
-rw-r--r--  1 root root       3028 Apr 26  2018
   adduser.conf
drwxr-xr-x  2 root root      12288 Sep  2 19:58
   alternatives
drwxr-xr-x  8 root root       4096 Aug 30 06:56
   apache2
drwxr-xr-x  3 root root       4096 Apr 26  2018 apm
drwxr-xr-x  3 root root       4096 Jan 28  2019
   apparmor
drwxr-xr-x  9 root root       4096 Sep  2 08:25
   apparmor.d
drwxr-xr-x  3 root root       4096 Jul  9 06:12 apport
drwxr-xr-x  7 root root       4096 May 23 09:04 apt
```

3.6 Informações sobre os arquivos e diretórios

O uso do formato longo de listagens (opção l) apresenta diversas informações importantes. Quando o primeiro símbolo é um traço (**-**) significa que é um arquivo.

Listagem 3.4: Símbolo -

```
vivas@musashi:/etc$ ls -l /etc/hosts
-rw-r--r-- 1 root root 293 Mar 25 19:20 /etc/hosts
```

Se for um diretório iremos encontrar o símbolo **d** como apresentado na Listagem 3.5.

Listagem 3.5: Símbolo d

```
drwxr-xr-x@ 12 root wheel 408 24 Out 17:38 usr
```

Pode ser também um link quando a primeira letra é um **l** como apresentado na Listagem 3.6.

Listagem 3.6: Símbolo l

```
lrwxr-xr-x@ 1 root wheel 11 24 Out 17:11 tmp ->
   private/tmp
```

3.7 Obtendo informações sobre diretórios

Se você deseja obter informações sobre um determinado diretório, mas não quer listar o conteúdo, utilize a opção **ls -ld**. A Listagem 3.7 apresenta o resultado do comando aplicado ao diretório /**etc**.

Listagem 3.7: Obtendo informações sobre diretórios

```
vivas@musashi:/etc$ ls -ld /etc
drwxr-xr-x 129 root root 12288 Sep  4 06:53 /etc
```

3.8 Listando Arquivos Ocultos

Alguns arquivos ficam ocultos nos diretórios e só são apresentados quando utilizamos os comandos **ls -a** ou **ls –all**, Listagem 3.8. Este comando lista todos os arquivos inclusive os ocultos.

Listagem 3.8: Listando Todos os Arquivos Inclusive os Ocultos

```
drwxr-xr-x  2 vivas vivas     4096 May 23 09:03  .vim
```

```
-rw-------  1 root   root     11066 May 23 09:04  .
   viminfo
-rw-rw-r--  1 vivas  vivas      253 Feb 14  2019  .wget
   -hsts
-rw-------  1 vivas  vivas       54 Jun 27 19:21  .
   Xauthority
drwx------  2 vivas  vivas     4096 Jun 27 19:21  .xpra
```

Caso desejemos listar apenas os arquivos ocultos, poderemos utilizar um dos seguintes comandos exemplificados na Listagem 3.9.

Listagem 3.9: Listando Apenas os Arquivos Ocultos

```
vivas@musashi:~$ ls -ld .?*
drwxr-xr-x 12 root  root   4096 Aug 26 17:21 ..
drwxrwxr-x  3 vivas vivas  4096 Mar 26 16:18 .anaconda
drwxrwxr-x  3 vivas vivas  4096 Mar 26 16:18 .astropy
-rw-------  1 vivas vivas 34405 Sep  3 17:13 .
   bash_history

vivas@musashi:~$ ls -a | egrep '^\.'
.
..
.anaconda
.astropy
.bash_history
.bash_logout
.bashrc
.bashrc-anaconda3.bak
.cache
.cmake
```

3.9 Classificando Arquivos e Diretórios

Para classificar os arquivos (/ para diretórios, * para executáveis, @ para links simbólicos, | para FIFOS e = para sockets) utilize a Listagem 3.10.

Listagem 3.10: Classificando Arquivos e Diretórios

```
vivas@musashi:/etc$ ls -F
cloud/                 libibverbs.d/        resolv.
   conf@
console-setup/         libnl-3/             rmt*
```

3.10 Imprimindo Informações sobre o Tamanho dos arquivos.

Para imprimir os arquivos com o tamanho em blocos, Listagem 3.11.

Listagem 3.11: Tamanho em Blocos

```
vivas@musashi:/usr/local$ ls -s
total 724148
636476 Anaconda3-5.2.0-Linux-x86_64.sh       4 julia
   -1.1.1
     4 bin                               87628 julia
        -1.1.1-linux-x86_64.tar.gz
     4 etc                                   4 lib
     4 games                                 0 man
     4 include                               4 sbin
     4 julia                                 4 share
     4 julia-1.1.0                           4 src
```

Para imprimir o tamanho em Kilobytes, Megabytes ou Terabytes utilize o código da Listagem 3.12.

Listagem 3.12: Tamanho dos Arquivos

```
drwxr-xr-x 13 root  root  4.0K Jul  5 14:16 .
drwxr-xr-x 10 root  root  4.0K Feb 13  2019 ..
-rwxr-xr-x  1 root  root  622M Feb 14  2019 Anaconda3
   -5.2.0-Linux-x86_64.sh
drwxr-xr-x  2 root  root  4.0K Sep  2 14:13 bin
drwxr-xr-x  3 root  root  4.0K Apr 26 16:18 etc
drwxr-xr-x  2 root  root  4.0K Apr 26  2018 games
drwxr-xr-x  2 root  root  4.0K Apr 26  2018 include
drwxr-xr-x 16 root  root  4.0K Feb  6  2019 julia
drwxr-xr-x  7 vivas vivas 4.0K Jan 21  2019 julia
   -1.1.0
drwxr-xr-x  7  1337  1337 4.0K May 16 02:56 julia
   -1.1.1
-rw-rw-r--  1 vivas vivas  86M May 16 21:27 julia
   -1.1.1-linux-x86_64.tar.gz
drwxr-xr-x  7 root  root  4.0K Mar 26 16:31 lib
lrwxrwxrwx  1 root  root     9 Apr 26  2018 man ->
   share/man
drwxr-xr-x  2 root  root  4.0K Apr 26  2018 sbin
drwxr-xr-x 12 root  root  4.0K Feb 14  2019 share
drwxr-xr-x  2 root  root  4.0K Apr 26  2018 src
```

Pode-se usar uma composição de opções **ls -alF**.

Listagem 3.13: Combinando Opções do Comando ls

```
vivas@musashi:/usr/local$ ls -alF
total 724156
drwxr-xr-x 13 root  root       4096 Jul  5 14:16 ./
drwxr-xr-x 10 root  root       4096 Feb 13  2019 ../
-rwxr-xr-x  1 root  root  651745206 Feb 14  2019
   Anaconda3-5.2.0-Linux-x86_64.sh*
drwxr-xr-x  2 root  root       4096 Sep  2 14:13 bin/
drwxr-xr-x  3 root  root       4096 Apr 26 16:18 etc/
drwxr-xr-x  2 root  root       4096 Apr 26  2018 games
   /
drwxr-xr-x  2 root  root       4096 Apr 26  2018
   include/
drwxr-xr-x 16 root  root       4096 Feb  6  2019 julia
   /
drwxr-xr-x  7 vivas vivas      4096 Jan 21  2019 julia
   -1.1.0/
drwxr-xr-x  7  1337  1337      4096 May 16 02:56 julia
   -1.1.1/
-rw-rw-r--  1 vivas vivas  89727957 May 16 21:27 julia
   -1.1.1-linux-x86_64.tar.gz
drwxr-xr-x  7 root  root       4096 Mar 26 16:31 lib/
lrwxrwxrwx  1 root  root          9 Apr 26  2018 man
   -> share/man/
drwxr-xr-x  2 root  root       4096 Apr 26  2018 sbin/
drwxr-xr-x 12 root  root       4096 Feb 14  2019 share
   /
drwxr-xr-x  2 root  root       4096 Apr 26  2018 src/
```

Algumas vezes você deseja visualizar o diretório e não o conteúdo do mesmo. Para isto, basta usar a opção -d:

Listagem 3.14: Visualizando Informações sobre o Diretório

```
vivas@musashi:/usr/local$ ls -ld /usr/
drwxr-xr-x 10 root root 4096 Feb 13  2019 /usr/
```

3.11 Listando Recursivamente

Para listar recursivamente o conteúdo de um diretório e seus sub-diretórios utilize o comando da Listagem 3.15. Para interromper o comando digite <**CONTROL+C**>.

Listagem 3.15: Listando Recursivamente

```
vivas@musashi:/usr/local$ ls -R /etc/
/etc/:
acpi                        inputrc
   popularity-contest.conf
adduser.conf                iproute2                    profile
alternatives                iptables                    profile.
   d
apache2                     iscsi
   protocols
```

3.12 Navegando em Diretórios

Use o comando **cd** para a navegação nos diretórios do sistema de arquivo do Linux. Para ir ao seu diretório home (seu diretório padrão de trabalho), basta digitar o comando da Listagem 3.16. Se o usuário for vivas, então para ir para o diretório é necessário digitar o caminho completo. O comando para navegar nos diretórios é o **cd**

Listagem 3.16: Comando cd

```
$ cd /home/vivas
```

Para ir rapidamente ao seu diretório home, apenas digite:

Listagem 3.17: Atalho para o Diretório Raiz do Usuário

```
$ cd ~
```

O símbolo ∼ é expandido pelo interpretador como */home/seu_usuario* ao executar o comando. Para verificar isso, experimente:

Listagem 3.18: Significado de ∼

```
vivas@musashi:/usr/local$ echo ~
/home/vivas
```

Alternativamente pode ser usada uma forma reduzida que produzirá o mesmo efeito, ou seja, o comando **cd** sozinho, Listagem 3.19.

Listagem 3.19: Comando cd Sem Opções

```
$ cd
```

Os diretórios . e .. referem-se ao diretório corrente e diretório-pai, respectivamente. Esses diretórios podem ser usados em sintaxes de comandos, assim como o ∼ também. Exemplos:

- **ls -la .**
- **ls -F ..**
- **cd ..**
- **cd ./dir1/dir2**

Para retornar para o diretório-pai basta usar o código da Listagem 3.20.

Listagem 3.20: Retornando ao Diretório do Usuário

```
$ cd  /home/usuario

$ cd ..
```

3.13 Comando pwd

Sempre que você está no terminal do sistema, você está sempre dentro de algum diretório. Para saber qual é a sua localização atual, você poderá utilizar o comando **pwd** como na Listagem 3.21.

Listagem 3.21: Comando pwd

```
vivas@musashi:/usr/local$ pwd
/usr/local
```

Todo diretório possui dois arquivos especiais cujos nomes consistem em um ou dois pontos: '.' ou '..'. Estes designam o diretório corrente e o diretório pai, respectivamente.

Ao designar um arquivo, subentende-se que está sendo referenciado o arquivo no diretório corrente. Outra maneira é especificar o arquivo com o caminho completo, como ilustrado na Listagem 3.22.

Listagem 3.22: Outro Exemplo do Comando pwd

```
$ pwd
/home/john
$ ls test.txt
$ ls /home/john/test.txt
$ ls ~/test.txt
```

3.14 Copiando Arquivos

Ao usar o comando **cp** (**copy**) , pode-se efetuar cópias de arquivo ou grupo de arquivos, bem como diretórios inteiros. O comando precisa de dois argumentos: o arquivo original e o destino. A sintaxe geral é apresentada na Listagem 3.23.

Listagem 3.23: Sintase do Comando cp

```
$ cp <opcoes> </localfonte/arquivo> </localdestino/>
```

3.14.1 Copiando Arquivo para Diretório

Para copiar um arquivo para um diretório utilize o comando da Listagem 3.24. Neste caso, o arquivo está no diretório corrente e será copiado para o diretório /home/pedro/documentos.

Listagem 3.24: Copiando Arquivo para Diretório

```
$ cp code.c /home/pedro/documentos
```

A barra final como indicação de diretório de destino é essencial. Caso não seja colocado, o sistema interpretará o último elemento do caminho de destino como sendo um nome de arquivo. Assim, ao copiar o arquivo code.c (exemplo acima) ele teria seu nome alterado para seu_usuário.

Suponha que esteja no diretório /home/pedro/programas e precise copiar o arquivo teste.c do diretório /home/pedro/aulas para /home/pedro/testes. Para isto, você vai precisar de passar o caminho completo do diretório ou o caminho relativo, como apresentado na Listagem 3.25.

Listagem 3.25: Copiando Arquivo para um Diretório

```
$ cp /home/pedro/aulas/teste.c /home/pedro/testes
$ cp ../aulas/teste.c ../testes
```

3.15 Copiando Múltiplos arquivos

Use o comando **cp** para copiar múltiplos arquivos para um diretório, podendo estar tais arquivos em locais diferentes como apresentado na Listagem 3.26.

Neste exemplo copiamos os arquivos *code.c* no diretório */home/pedro/-dir1/* e o arquivo *main.c* no diretório */home/pedro/dir2/* e finalmente o arquivo *teste.c* do diretório */home/pedro/dir3/* para o diretório */home/pedro/*. Pelo exemplo, podemos ver a flexibilidade do sistema, tendo o usuário total liberdade de definir quantos e quais arquivos devem ser copiados para o diretório de destino.

Listagem 3.26: Copiando Múltiplos Arquivos

```
$ cp /home/pedro/dir1/code.c /home/pedro/dir2/main.c /
   home/pedro/dir3/teste.c   /home/pedro/
```

Use o comando **cp** para copiar um arquivo para outro, Listagem 3.27. Dessa forma, o sistema não interpelará, i.e., caso haja um outro arquivo com o nome do arquivo a ser criado ele será sobrescrito sem nenhum impedimento.

O sistema admitirá que você, usuário, sabe o que está fazendo. Para se efetuar a mesma ação com a necessidade de confirmação, você deve usar a opção -i para alertá-lo, caso seja necessário sobrescrever algum arquivo.

Listagem 3.27: Copiando um Arquivo em Outro

```
$ cp code.c main.c
```

A Listagem 3.28 apresenta o comando copy com confirmação (opção -i). Neste exemplo, cria o arquivo main.c caso não exista e lhe pede confirmação da ação, caso seja necessário sobrescrever um arquivo já existente.

Listagem 3.28: Copiando com a Opção -i

```
$ cp -i code.c main.c
cp: overwrite 'main.c'?
```

3.16 Copiando Diretórios e Sub-diretórios

Use a opção -r para copiar hierarquias inteiras de arquivos e sub-diretórios (inclusive arquivos ocultos) como apresentado na Listagem 3.29. Neste caso, iremos copiar o diretório /home/maria/teste para o diretório /home/maria/temp.

Listagem 3.29: Copiando Diretórios

```
$ cp -r /home/maria/teste /home/maria/temp
```

3.17 Renomeando Arquivos

O comando **rename** (**rename**) é utilizado para renomear um ou mais arquivos. A sintaxe geral é apresentada na Listagem 3.30, através de um exemplo em que o arquivo *teste.txt* será renomeado para *teste2.txt*.

Listagem 3.30: Renomeando Arquivos com rename

```
$ rename teste.txt teste2.txt
```

3.18 Criando um Arquivo Vazio com touch

O comando **touch** (**touch**) é utilizado para criar um ou mais arquivos vazios. A sintaxe geral é apresentada na Listagem 3.31, através do exemplo que criará o arquivo vazio *teste.txt*.

Listagem 3.31: Criando Arquivos com touch

```
$ touch teste.txt
```

3.19 Criando arquivos ou diretórios temporários

Em um *script*, muitas vezes precisamos armazenar resultados intermediários em arquivos. Para tanto recorremos a arquivos temporários. Poderíamos criar um arquivo qualquer e utilizá-lo como um arquivo temporário, entretanto, pode ser necessário criar diversos arquivos desses. Para resolver este problema podemos utilizar o comando **mktemp** . A Listagem 3.32 mostra um exemplo simples de utilização.

Listagem 3.32: Criando arquivo temporário

```
$ TMPFILE=$(mktemp)
$ echo $TMPFILE
/tmp/tmp.M4gNs2xM4E
$ echo "temp file" > $TMPFILE
$ cat $TMPFILE
temp file
$ rm $TMPFILE
$ ls $TMPFILE
ls: cannot access '/tmp/tmp.M4gNs2xM4E': No such file
   or directory
```

3.20 Apagando Arquivos

Para apagar um arquivo, utilizamos o comando **rm** como mostrado na Listagem 3.33. Vamos supor que precisamos apagar o arquivo *teste.c*.

Listagem 3.33: Apagando um Arquivo

```
$ rm teste.c
```

3.20.1 Apagando Múltiplos Arquivos

Vamos supor que precisamos apagar todos os arquivos com a extensão **.c**. Utilize a Listagem 3.34.

Listagem 3.34: Apagando Múltiplos Arquivos

```
$ rm *.c
```

3.21 Apagando um Diretório

Quer apagar um diretório inteiro? Vamos utilizar o comando **rm** como na Listagem 3.35. Vamos supor que precisamos apagar o sub-diretório músicas, para tanto, será necessário apagar os arquivos dentro deste diretório e o próprio diretório. Podemos passar o argumento `-r` ou `-R` para que sejam apagados recursivamente os arquivos e subdiretórios dentro do diretório que desejamos remover e também o próprio diretório a ser removido.

Listagem 3.35: Apagando Diretório

```
$ rm -R musicas/
```

Em um diretório com muitos arquivos e subdiretórios, é possível que encontremos vários arquivos com proteção para escrita. O comando **rm** irá perguntar ao usuário se deve apagar acada um deles. Para evitar isso, você pode utilizar o argumento **-f** (ou –**force**) para não precisar responder 'sim' a todas as peguntas, ou seja, irá forçar a remoção dos arquivos sem perguntar ao usuário.

3.22 Apagando Diretório com rmdir

O comando **rmdir** pode ser utilizado para apagar diretórios como realizado na Listagem 3.36, onde é criado o diretório *teste* e em seguido é apagado.

Listagem 3.36: Apagando Diretório com rmdir

```
mkdir teste
rmdir teste
```

3.23 Nomes de arquivos

Um arquivo é uma coleção de dados relacionados que compõem uma unidade, um bloco contíguo de informação que é armazenado, por exemplo, em um disco rígido, pendrives ou um disco ótico, etc. Os nomes são dados aos arquivos para permitir ao usuário identificar e acessar de forma simples estes dados. Os nomes de arquivos e diretórios são então apenas uma conveniência para os usuários.

Os nomes de arquivos em Linux podem conter qualquer caractere, exceto a barra /, que é reservada como separador de diretórios e para identificar o diretório raiz, e o caractere nulo (*null*) que é utilizado como terminação de um segmento. Espaços e caracteres especiais são permitidos, embora devam ser evitados para não causar incompatibilidade entre softwares.

Tipicamente os nomes de arquivos contém apenas caracteres alfanuméricos (em caixa baixa, majoritariamente), sublinhado (*underscore*), hifens e

ponto final. Outros caracteres são evitados, pois possuem significado especial para o *shell* e podem complicar a vida do usuário.

Embora os sistemas Unix não requeiram a utilização de extensão de arquivo (uma sequência no final do nome do arquivo constituída de ponto final seguido de um a três caracteres), pode ser conveniente e útil a sua utilização, tornando simples a tarefa de identificar tipos de arquivo à uma primeira vista.

Os diretórios são vistos como apenas um tipo especial de arquivo e por isto, as convenções de utilização de nomes para eles também são as mesmas.

A definição da interface POSIX, a fim de garantir compatibilidade e interoperabilidade entre sistemas operacionais, especifica algumas regras para a nomenclatura de arquivos. Os caracteres que podem ser utilizados são aqueles listados na Listagem 3.37. O tamanho máximo do nome de um arquivo deve ser 14. Desta forma, existem 24407335764928225040435790 combinações para compor nomes de arquivos com até 14 caracteres utilizando os 65 caracteres na Listagem 3.37.

Listagem 3.37: Caracteres permitos pelo POSIX

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
0123456789._-
```

Outras duas regras, para não criar distúrbios, são: não iniciar o nome de um arquivo com ponto o traço. Os aquivos cujo nome começam com ponto são os arquivos ocultos. Arquivos iniciados com traço podem ser entendidos como um argumento para muitos comandos.

Alguns arquivos possuem nomes tão diferentes que os tornam difíceis de serem apagados. Um exemplo são os arquivos cujos nomes se iniciam com um hífen. Se o arquivo possui espaços ou caracteres passíveis de interpretação pelo Linux (eg, * ou -) deve-se colocar aspas simples ao referenciá-lo.

3.23.1 Barra invertida

A barra invertida é utilizada para se possibilitar o uso de mais de uma linha na escrita de uma sintaxe ou de um encadeamento de comandos muito extenso. O interpretador tratará as linhas terminadas com \ como se fossem uma única linha, justamente para dar uma maior flexibilidade ao usuário nas suas definições. Um sinal de maior é incluído pelo interpretador para mostrar a continuidade da linha anterior.

3.24 Criando Diretório

O comando **mkdir** (*make directory*) cria um ou mais sub-diretórios. A Listagem 3.38 apresenta o código de criação do diretório teste.

Listagem 3.38: Criando Diretório com mkdir

```
$ mkdir teste
```

3.25 Criando Múltiplos Diretórios

Quer criar vários diretórios de uma única vez. Utilize o comando **mkdir** como apresentado na Listagem 3.39. Neste exemplo são criados os diretórios *tempdir1*, *tempdir2* e *tempdir3*.

Listagem 3.39: Criando Múltiplos Diretórios

```
$ mkdir tempdir1 tempdir2 tempdir3
```

3.26 Criar Hierarquia de Diretórios

Se você precisa criar uma hierarquia de sub-diretórios de uma única vez, basta usar a opção `-p` como na Listagem 3.40, desta forma todos os diretório ha hierarquia serão criados, caso não existam.

Listagem 3.40: Criando Árvore de Diretórios

```
$ mkdir -p temp/temp1/tempdir
```

3.27 Links

Sistemas Operacionais possuem um recurso que permite dar apelidos para arquivos e ou diretórios. Este recurso é denominado atalho e é implementado pelo comando **ln** . O comando **ln** permite que isto seja realizado e denominamos estes apelidos de **links**. Os **links** podem ser simbólicos (**soft links**) ou (**hard links**). A Listagem 3.41 apresenta a listagem do diretório */etc* onde o símbolo `l` indica que *blkid.tab* é um link.

Listagem 3.41: Links versus Arquivos e Diretórios

```
$ ls -l /etc
total 1140
drwxr-xr-x  3 root root    4096 Ago 20 14:59 acpi
...
-rw-r--r--  1 root root    2981 Ago 20 14:56 adduser.
   conf
...
lrwxrwxrwx  1 root root      15 Nov 21 19:08 blkid.tab
    -> /dev/.blkid.tab
...
```

Utilizamos links simbólicos para criar atalhos para arquivos ou diretórios. Por exemplo, para garantir que compatibilidade com um software antigo.

Vamos aprender a criar links simbólicos através do exemplo a seguir. O primeiro passo é criar um arquivo chamado *arq1* como na Listagem 3.42.

Listagem 3.42: Criando um Arquivo

```
$ touch arq1
$ echo 'Laranja' >> arq1
```

Agora vamos criar um link simbólico chamado *arq1-soft* para o arquivo *arq1* e listar os dois arquivos, Listagem 3.43.

Listagem 3.43: Criando um Link Simbólico

```
$ ln -s arq1 arq1-soft
$ ls -l arq1*
-rw-rw-r-- 1 vivas vivas 8 Jan  3 16:18 arq1
lrwxrwxrwx 1 vivas vivas 4 Jan  3 16:18 arq1-soft ->
   arq1
```

Agora vamos listar o conteúdo do arquivo *arq1-soft* como na Listagem 3.44. Como pode ser observado o conteúdo é o mesmo, pois o arquivo *arq1-soft* é um link para o arquivo *arq1*.

Listagem 3.44: Verificando o Conteúdo do Arquivo

```
$ cat arq1-soft
Laranja
```

Agora vamos mover o *arq1* para um novo arquivo chamado *arq2*. Após mover o arquivo vamos tentar listar o conteúdo do link simbólico como na Listagem 3.45.

Listagem 3.45: Movendo o Arquivo

```
$ mv arq1 arq2
$ cat arq1-soft
cat: arq1-soft: Arquivo ou diretorio nao encontrado
```

Vamos agora listar o arq1-soft e verificar que ele está apontando para o arquivo *arq1* que agora não existe mais como na Listagem 3.46

Listagem 3.46: Visualizando os Links

```
$ ls -l arq1-soft
lrwxrwxrwx 1 vivas vivas 4 Jan  3 16:18 arq1-soft ->
   arq1
```

3.27.1 Hard Links

Os links simbólicos (*soft links*) são diferentes dos hard links, pois os simbólicos referem-se a outros arquivos pelo nome. Os hard links fazem esta ligação pelo inode e por isto possuem o mesmo inode. Os links simbólicos fazem um atalho para o nome do arquivo e não para a posição do arquivo no disco.

Observações gerais

- Hard link apresenta em suas informações o mesmo tamanho do arquivo original.
- Link simbólico apresenta um tamanho reduzido, pois guarda o nome do arquivo original. O tamanho é tão grande quanto for o nome desse arquivo.
- Link simbólico possui inode distinto do arquivo original.
- Hard link possui o mesmo inode do arquivo original.

Vamos agora aprender a trabalhar com hard links. Um hard link aponta para o inode, isto é, para a posição do disco e não para o nome do arquivo (como os soft-link). O primeiro passo é criar um arquivo e preenchê-lo com alguma informação como na Listagem 3.47.

Listagem 3.47: Hard Links - Passo 1

```
$ touch arq1
$ echo "Laranja" >> arq1
$ cat arq1
Laranja
```

Agora vamos criar um hard link e listar o conteúdo do mesmo como na Listagem 3.48.

Listagem 3.48: Hard Links - Passo 2

```
$ ln arq1 arq1-hard
$ cat arq1-hard
Laranja
```

Liste os arquivos criados e verifique o que aconteceu como na Listagem 3.49.

Listagem 3.49: Hard Links - Passo 3

```
$ ls -l arq1*
-rw-rw-r-- 2 vivas vivas 8 Jan 3 16:45 arq1
-rw-rw-r-- 2 vivas vivas 8 Jan 3 16:45 arq1-hard
```

Agora vamos mover o *arq1* para *arq2* e verificar o que aconteceu como na Listagem 3.50.

Listagem 3.50: Hard Links - Passo 4

```
$ mv arq1 arq2
$ cat arq1-hard
Laranja
$ ls -l arq1-hard
-rw-rw-r-- 2 vivas vivas 8 Jan 3 16:45 arq1-hard
```

Como verifica-se no exemplo, o *arq1-hard* continuou existindo, pois quando um hard link é criado ele aponta para o inode (posição no disco) e não para o nome do arquivo.

Symlinks são distintamente diferentes de arquivos comuns. Portanto, podemos distinguir um symlink do arquivo original para o qual ele aponta. Symlinks podem referenciar (apontar) qualquer tipo de arquivo. Symlinks referem-se a nomes e, daí podem apontar para arquivos localizados em outro sistema de arquivos. Se você renomeia ou apaga o arquivo original apontado pelo symlink, o symlink se rompe. Symlinks podem ocupar espaço em disco adicional para armazenar o nome do arquivo apontado.

Hard links múltiplos, com nomes diversos para um mesmo arquivo, são ilimitados. Hard links trabalham com o número do inode e, portanto, eles só podem atuar dentro de um único sistema de arquivo. Ao copiar ou deletar o arquivo original apontado pelo hard link não se tem nenhum efeito sobre o hard link. Hard links necessitam somente de espaço suficiente para o armazenamento de uma entrada de diretório.

Capítulo 4

Comandos para Manipulação de Arquivos Texto

Sumário

4.1 Comando echo

O commando mais simples é o **echo**. Ele apenas exibe uma linha de texto, ou seja, ecoa.

Listagem 4.1: Comando echo

```
$ echo 'olá Alessandro!'
olá Alessandro!
```

Caso queira suprimir a quebra de linha ao final da string, basta usar o parâmetro **-n** e para habilitar a interpretação da contra-barra (\) utilize o parâmetro **-e**.

Listagem 4.2: Comando echo

```
$ echo -en 'olá\tAlessandro!'
olá     Alessandro!
```

4.2 Comando cat

Outro comando simples que é muito utilizado é o **cat**. Este comando concatena arquivos e imprime o resultado na saída padrão. Caso seja passado como argumento apenas o nome de um único arquivo, não haverá outro para concatenar e o programa envia o arquivo para a saída padrão.

Listagem 4.3: Comando cat

```
$ cat cidades.txt
Abadia de Goias (GO)
Abadia dos Dourados (MG)
Abadiania (GO)
...
```

Vamos agora criar dois arquivos. O primeiro possui os números de 1 a 5 e o segundo os números de 6 a 10 (iremos para tanto utilizar o comando **seq**, veja a secção 4.4). Em seguida, vamos utilizar o comando **cat** para concatenar os dois arquivos e jogar o resultado na saída padrão.

Listagem 4.4: Comando cat com dois arquivos

```
$ seq 1 5 > arq1.txt
$ seq 6 10 > arq2.txt
$ cat arq1.txt arq2.txt
1
2
3
```

```
4
5
6
7
8
9
10
```

4.3 Comando cut

Precisa cortar caracteres de uma frase? Vamos utilizar o comando **cut**. Neste exemplo, Listagem 4.5, vamos imprimir somente os caracteres 1 até 10 da palavra "alessandro vivas andrade". O comando **echo** serve para imprimir texto na tela do interpretador de comandos.

Listagem 4.5: Comando cut

```
$ echo 'alessandro vivas andrade' | cut -c 1-10
alessandro
```

Podemos utilizar o comando **cut** para listar os usuários logados no sistema. Veja o exemplo da Listagem 4.6.

Listagem 4.6: Usando o comando cut para listar os usuários logados

```
$ who | cut -c1-8 | sort | uniq
leoca
vivas
```

Outro problema recorrente é separação de valores em um arquivo. Na Listagem 4.7 temos um arquivo com estado com dois caracteres e após o nome da cidade. Vamos salvar este arquivo com o nome cidades.txt.

Listagem 4.7: Arquivo com Cidades

```
SP Sao Paulo
SP Campinas
MG Belo Horizonte
MG Diamantina
MG Lavras
MG Bom Sucesso
RJ Rio de Janeiro
```

Partindo da Listagem 4.7, vamos imprimir somente os dois primeiros caracteres de cada linha de um arquivo, Listagem 4.8. Para isto, vamos utilizar o símbolo | (pipe) neste comando. Este símbolo tem como função separar comandos no shell. O resultado do primeiro comando é direcionado para o segundo comando.

Listagem 4.8: Separando dados de um Arquivo

```
$ cat cidades.txt | cut -c 1-2
SP
SP
MG
MG
MG
MG
RJ
```

4.4 Comando seq

O comando **seq** é utilizado para gerar uma sequência de números. Se passar como argumento apenas um número, será gerada uma sequência de 1 até o valor do argumento.

Listagem 4.9: Gerando uma sequência de números

```
$ seq 3
1
2
3
```

Se quiser gerar uma sequência que não comece do 1, deverá passar dois argumentos: o valor inicial e o valor final da sequência.

Listagem 4.10: Gerando uma sequência de números

```
$ seq 3 5
3
4
5
```

O padrão do comando **seq** é gerar uma sequência de números espaçados de 1 em 1. Você pode especificar o tamanho do passo, se desejar. Para tanto, deverá passar mais um argumento para o comando, totalizando 3 argumentos.

Listagem 4.11: Gerando uma sequência de números

```
$ seq 1.1 0.1 1.3
1,1
1,2
1,3
```

4.5 Comando expand

Imagine que agora nosso arquivo tenha tabulações, Listagem 4.12, separando os campos e precisamos converter as tabulações em caracteres. Para realizar esta tarefa utilize o comando **expand** . A Listagem 4.12 apresenta os dados originais, armazenados no arquivo *cidades.txt*.

Listagem 4.12: Listagem Original

```
SP   Sao Paulo:pedro
SP   Campinas:andre
MG   Belo Horizonte:marta
MG   Diamantina:lucas
MG   Lavras: rafael
MG   Bom Sucesso:lisa
RJ       Rio de Janeiro:alexandro
```

A Listagem 4.13 converte a tabulação do arquivo para 1 espaço. O número após t indica o número de caracteres que a tabulação será convertida.

Listagem 4.13: Tabulações Convertidas para 1 Espaço

```
$ cat cidades.txt | expand -t 1
SP Sao Paulo:pedro
SP Campinas:andre
MG Belo Horizonte:marta
MG Diamantina:lucas
MG Lavras: rafael
MG Bom Sucesso:lisa
RJ Rio de Janeiro:alexandro
```

Na Listagem 4.14 a tabulação foi convertida para 10 caracteres.

Listagem 4.14: Convertendo Tabulações

```
cat cidades.txt | expand -t 10
SP        Sao Paulo:pedro
SP        Campinas:andre
MG        Belo Horizonte:marta
MG        Diamantina:lucas
MG        Lavras: rafael
MG        Bom Sucesso:lisa
RJ        Rio de Janeiro:alexandro
```

Podemos fazer este tipo de substituição utilizando um comando mais genérico, o comando **tr**, como veremos na proxima seção.

4.6 Comando tr

O comando **tr** é utilizado para efetuar substituições (ou tradução) e apagar caracteres.

A Listagem 4.15 apresenta um exemplo em que utilizaremos o comando **tr** para substituir tabulações por um único espaço simples.

Listagem 4.15: Convertendo tabulações em espaço simples

```
$ cat cidades.txt
SP      Sao   Paulo
SP      Campinas
MG      Belo     Horizonte
MG      Diamantina
MG      Lavras
MG      Bom         Sucesso
RJ      Rio de Janeiro
$ cat cidades.txt | tr '\t' ' '
SP Sao   Paulo
SP Campinas
MG Belo     Horizonte
MG Diamantina
MG Lavras
MG Bom         Sucesso
RJ Rio de Janeiro
```

Se além disso, queremos substituir as múltiplas ocorrências de espaços por um único espaço, podemos proceder como ilustrado na Listagem 4.16.

Listagem 4.16: Convertendo tabulações em espaço simples e removendo múltiplas ocorrências de espaços

```
$ cat cidades.txt | tr '\t' ' ' | tr -s ' '
SP Sao Paulo
SP Campinas
MG Belo Horizonte
MG Diamantina
MG Lavras
MG Bom Sucesso
RJ Rio de Janeiro
```

O comando **tr** pode ser utilizado para realizar diversos outros tipos de substituições ou para apagar caracteres indesejáveis. Veremos abaixo alguns exemplos.

Listagem 4.17: Convertendo MAIÚSCULA em minúsculas

```
$ tr 'A-Z' 'a-z' < cidades.txt
sp      sao paulo
sp      campinas
mg      belo horizonte
mg      diamantina
mg      lavras
mg      bom sucesso
rj      rio de janeiro
```

Uma outra forma de realizar a substituição de maiúsculas por minúsculas é apresentada na Listagem 4.18.

Listagem 4.18: Outra forma de converter MAIÚSCULA em minúsculas

```
$ cat cidades.txt | tr [:upper:] [:lower:]
sp      sao paulo
sp      campinas
mg      belo horizonte
mg      diamantina
mg      lavras
mg      bom sucesso
rj      rio de janeiro
```

O exemplo da Listagem 4.19 ilustra como transformar espaços em branco (incluindo aqui tabulações e quebras de linhas) em uma quebra de linha. Para tanto, utilizaremos `[:space:]` para designar qualquer um dos caracteres: espaço, tabulação e quebra de linha.

Listagem 4.19: Transformar espaços em quebra de linha

```
$ cat cidades.txt | tr -s [:space:] '\n'
SP
Sao
Paulo
SP
Campinas
MG
Belo
Horizonte
MG
Diamantina
MG
Lavras
MG
Bom
Sucesso
```

```
RJ
Rio
de
Janeiro
```

Podemos utilizar o comando **tr** para substituir um conjunto de caracteres. Para tanto, será considerada a ordem em que eles parecem. No exemplo apresentado na Listagem 4.20, iremos substituir `{` por `(` e `}` por `)`.

Listagem 4.20: Substituir chaves por parênteses

```
$ echo 'a{1}, b{2}' | tr '{}' '()'
a(1), b(2)
```

Caso deseje remover os algarismo de 0 a 9, basta utilizar uma das duas formas ilustradas na Listagem 4.21.

Listagem 4.21: Duas maneiras para se remover dígitos

```
$ cat numeros.txt
1 Diamantina
2 Datas
3 Gouveia
4 Mendanha
5 Curvelo
$ cat numeros.txt | tr -d [:digit:]
 Diamantina
 Datas
 Gouveia
 Mendanha
 Curvelo
$ cat numeros.txt | tr -d '0-9'
 Diamantina
 Datas
 Gouveia
 Mendanha
 Curvelo
```

Algumas vezes pode ser mais simples definir por exclusão. Suponha que você queira remover apenas as consoantes do exemplo anterior, basta utilizarmos o parâmetro `-c` para obter o complemento. Veja o exemplo na Listagem 4.22.

Listagem 4.22: Usando o complemento para remover as consoantes

```
$ tr -dc 'aeiouAEIOU' < numeros.txt
iaaiaaaoueiaeaaueo
```

4.7 Comando fmt

O comando **fmt** é usado para formatar arquivos texto. Usado para organizar as palavras (grupos de caracteres) de um arquivo para uma forma consistente, i.e., com um número de caracteres por linha definido.

Seja o seguinte texto, sem delimitação de caracteres: "Antes da chegada dos colonizadores portugueses, no século XVI (os primeiros relatos dão conta de expedições que subiram o Rio Jequitinhonha e São Francisco) , Diamantina, como toda a região do atual estado de Minas Gerais, era ocupada por povos indígenas do tronco linguístico". Vamos supor que precisamos formatar o texto e limitar 10 caracteres por linha. Para fazer isto, utilizamos o comando **fmt** da Listagem 4.23.

Listagem 4.23: Formatando Linhas com o Comando fmt

```
$ fmt -w10 texto.txt
Antes da
chegada
dos
colonizadores
portugueses,
no
...
continua
```

Observe no resultado do comando **fmt** acima que algumas linhas possuem largura superior a 10. O comando **fmt** preserva as palavras. Se você quiser quebrar a sequência de caracteres para rigidamente não ultrapassar o limite escolhido, deverá utiliza o comando **fold** (veja a secção 4.8).

4.8 Comando fold

O comando **fold** é utilizado para limitar o comprimento das linhas. Os caracteres que excederem o limite serão arrolados para a próxima linha. O exemplo na Listagem 4.24 limita a largura a 10 caracteres do mesmo texto utilizado na Listagem 4.23. Compare os resultados.

Listagem 4.24: Usando o comando fold para limitar a largura do texto

```
$ fold -w 10 texto.txt
Antes da c
hegada dos
 colonizad
ores portu
gueses, no
 século X
```

```
VI (os pri
...
continua
```

Caso você queria que a quebra ocorra preferencialmente nos espaços, poderá utilizar o parâmetro **-s**. Veja o exemplo na Listagem 4.25.

Listagem 4.25: Usando o comando fold com quebra preferencial nos espaços

```
$ fold -s -w 20 texto.txt
Antes da chegada
dos colonizadores
portugueses, no
século XVI (os
primeiros relatos
...
continua
```

4.9 Comando grep

O comando **grep** pode ser utilizar para procurar padrões em arquivos texto. Ele pode ser utilizado sozinho ou em conjunto com outros comandos. Vamos usar a Listagem 4.26 como exemplo. Suponha que você deseja encontrar em um arquivo um determinado padrão, como por exemplo, alexandro.

Listagem 4.26: Listagem para uso do grep

```
SP  Sao Paulo:pedro
SP  Campinas:andre:(074) 2233-1111
MG Belo Horizonte:marta:010 1222 0011
MG Diamantina:lucas:2
MG Lavras: rafael
MG Bom Sucesso:lisa
RJ   Rio de Janeiro:alexandro
```

Se os dados estiverem em um arquivo com nome *lista.txt* use a Listagem 4.27.

Listagem 4.27: Filtrando com grep

```
$ cat lista.txt | grep alexandro
RJ   Rio de Janeiro:alexandro
```

O operador | é chamado de pipe e é utilizado para concatenar um ou mais comandos.

Podemos utilizar também expressões regulares para buscar resultados usando o **grep**. No exemplo da Listagem 4.28, vamos buscar as linhas onde aparecem números.

Listagem 4.28: Usando regexp no grep

```
$ grep '[0-9]' lista.txt
SP  Campinas:andre:(074) 2233-1111
MG Belo Horizonte:marta:010 1222 0011
MG Diamantina:lucas:2
```

Podemos usar uma expressão regular para selecionar apenas as linhas onde aparece uma sequência de 2 ou mais números. Veja a Listagem 4.29.

Listagem 4.29: Usando regexp no grep

```
$ grep '[0-9]\{2,\}' lista.txt
SP  Campinas:andre:(074) 2233-1111
MG Belo Horizonte:marta:010 1222 0011
```

Outro exemplo bem interessante é buscar por números de telefones utilizando o **grep**. A Listagem

Listagem 4.30: Usando regexp e grep para encontrar as linhas contendo números de telefone

```
$ grep '\(([0-9]\{3\})\|[0-9]\{3\}\)[ -]\?[0-9]\{4\}[
   -]\?[0-9]\{4\}' lista.txt
SP  Campinas:andre:(074) 2233-1111
MG Belo Horizonte:marta:010 1222 0011
```

No exemplo anterior, estamos buscando por um padrão de três números opcionalmente entre parênteses, seguido (com espaço ou hífen separando, opcionalmente) de duas sequências de quatro números separadas por espaço ou hífen.

Listagem 4.31: Usando regexp e grep para encontrar as linhas contendo números de telefone (inclusive com prefixo internacional)

```
$ grep '\(\+[0-9]\{1,3\}\)\?[ -]\?\(([0-9]\{3\})
   \|[0-9]\{3\}\)[ -]\?[0-9]\{4\}[ -]\?[0-9]\{4\}'
   lista.txt
```

Existem outras alternativas para este comando como **egrep** e o **fgrep** .

4.10 Comando egrep

O comando **egrep** pode ser utilizado para criar expressões regulares. Uma aplicação simples seria buscar mais de uma palavra em um arquivo ou como resultado de um comando. Neste exemplo, vamos buscar duas palavras diferentes: firmware e amd no comando **dmesg**. A Listagem 4.32 apresenta o resultado do comando.

Listagem 4.32: Filtrando com egrep

```
$ dmesg | egrep "firmware|amd"
[    0.000000] Linux version 4.15.0-58-generic (
    buildd@lcy01-amd64-013) (gcc version 7.4.0 (Ubuntu
    7.4.0-1ubuntu1~18.04.1)) #64-Ubuntu SMP Tue Aug 6
    11:12:41 UTC 2019 (Ubuntu 4.15.0-58.64-generic
    4.15.18)
[    0.069054] Spectre V2 : Enabling Restricted
    Speculation for firmware calls
[    1.184190] acpi PNP0A08:00: PCIe AER handled by
    firmware
[    1.412723] acpi PNP0A08:01: PCIe AER handled by
    firmware
[    3.279447] GHES: APEI firmware first mode is
    enabled by APEI bit and WHEA _OSC.
[    5.866568] megaraid_sas 0000:01:00.0: firmware
    supports msix        : (0)
[    5.976880] megaraid_sas 0000:01:00.0: firmware
    crash dump   : no
```

4.11 Comando fgrep

O comando **fgrep** pode ser utilizado para filtrar caracteres especiais como apresentado na Listagem 4.33.

Listagem 4.33: Filtrando com fgrep

```
$ dmesg | fgrep "@"
[    0.000000] Linux version 4.15.0-58-generic (
    buildd@lcy01-amd64-013) (gcc version 7.4.0 (Ubuntu
    7.4.0-1ubuntu1~18.04.1)) #64-Ubuntu SMP Tue Aug 6
    11:12:41 UTC 2019 (Ubuntu 4.15.0-58.64-generic
    4.15.18)
[    0.000000] percpu: Embedded 46 pages/cpu @
    (ptrval) s151552 r8192 d28672 u262144
[    0.076000] smpboot: CPU0: Intel(R) Xeon(R) CPU
              E7530  @ 1.87GHz (family: 0x6, model: 0
    x2e, stepping: 0x6)
[    3.555080] device-mapper: ioctl: 4.37.0-ioctl
    (2017-09-20) initialised: dm-devel@redhat.com
[    6.333017] qla2xxx [0000:08:00.0]-00fc:3: ISP2432:
     PCIe (2.5GT/s x4) @ 0000:08:00.0 hdma+ host#=3 fw
    =8.07.00 (9496).
```

```
[    6.752541] qla2xxx [0000:08:00.1]-00fc:4: ISP2432:
     PCIe (2.5GT/s x4) @ 0000:08:00.1 hdma+ host#=4 fw
   =8.07.00 (9496).
[    7.168514] qla2xxx [0000:42:00.0]-00fc:5: ISP2432:
     PCIe (2.5GT/s x4) @ 0000:42:00.0 hdma+ host#=5 fw
   =8.07.00 (9496).
[    7.584714] qla2xxx [0000:42:00.1]-00fc:6: ISP2432:
     PCIe (2.5GT/s x4) @ 0000:42:00.1 hdma+ host#=6 fw
   =8.07.00 (9496).
[    8.000735] qla2xxx [0000:45:00.0]-00fc:7: ISP2432:
     PCIe (2.5GT/s x4) @ 0000:45:00.0 hdma+ host#=7 fw
   =8.07.00 (9496).
[    8.416916] qla2xxx [0000:45:00.1]-00fc:8: ISP2432:
     PCIe (2.5GT/s x4) @ 0000:45:00.1 hdma+ host#=8 fw
   =8.07.00 (9496).
[    8.832763] qla2xxx [0000:46:00.0]-00fc:9: ISP2432:
     PCIe (2.5GT/s x4) @ 0000:46:00.0 hdma+ host#=9 fw
   =8.07.00 (9496).
[    9.248608] qla2xxx [0000:46:00.1]-00fc:10: ISP2432
   : PCIe (2.5GT/s x4) @ 0000:46:00.1 hdma+ host#=10
   fw=8.07.00 (9496).
```

4.12 Comando head

O comando **head** é utilizado para imprimir as n linhas iniciais de um arquivo. Imagine um arquivo com o nome de *cidades.txt* com todas as cidades do Brasil. A Listagem 4.34 apresenta a lista impressa do arquivo *cidades.txt*.

Listagem 4.34: Imprime as Linhas Iniciais de um Arquivo

```
$ head cidades.txt
Abadia de Goias (GO)
Abadia dos Dourados (MG)
Abadiania (GO)
Abaete (MG)
Abaetetuba (PA)
Abaiara (CE)
Abaira (BA)
Abare (BA)
Abatia (PR)
Abdon Batista (SC)
```

Para imprimir as N linhas iniciais utilize o comando da Listagem 4.35. Neste exemplo, iremos imprimir as duas linhas iniciais.

Listagem 4.35: Imprime as Duas Linhas Iniciais de um Arquivo

```
$ head -2 cidades.txt
Abadia de Goias (GO)
Abadia dos Dourados (MG)
$ head -n 2 cidades.txt
Abadia de Goias (GO)
Abadia dos Dourados (MG)
```

Caso utilize o parâmetro com sinal negativo, o comando irá imprimir todas as linhas exceto as N linhas finais.

4.13 Comando tail

O comando **tail** é utilizado para imprimir as n linhas finais de um arquivo. Imagine um arquivo com o nome de *cidades.txt* com todas as cidades do Brasil. A Listagem 4.36 apresenta a lista impressa pelo comando **tail** para arquivo *cidades.txt*.

Listagem 4.36: Imprime as Linhas Finais de um Arquivo

```
$ tail cidades.txt
Xangri-la (RS)
Xanxere (SC)
Xapuri (AC)
Xavantina (SC)
Xaxim (SC)
Xexeu (PE)
Xinguara (PA)
Xique-Xique (BA)
Zabele (PB)
Zacarias (SP)
```

Para imprimir as N linhas finais utilize o comando da Listagem 4.37. Neste exemplo iremos imprimir as duas linhas finais.

Listagem 4.37: Imprime as Duas Linhas Finais de um Arquivo

```
$ tail -2 cidades.txt
Zabele (PB)
Zacarias (SP)
$ tail -n 2 cidades.txt
Zabele (PB)
Zacarias (SP)
```

4.14 Comando file

Os arquivos possuem nomes e usualmente adota-se a prática de utilizar como parte do nome a extensão, uma sequência de caracteres (usualmente 3 ou 4) após um ponto final, no fim do nome do aquivo. Assim entendemos que um arquivo chamado `lena.jpg` será uma imagem codificada no formato JPEG, embora possa não ser o caso (afinal é apenas o nome).

Para realmente determinar o tipo de um arquivo, podemos utiliza o comando **file**. O exemplo a seguir mostra a utilização do comando para determinar o tipo do arquivo `lena.jpg`.

Listagem 4.38: Determinando o tipo do arquivo `lena.jpg`

```
$ file lena.jpg
lena.jpg: JPEG image data, JFIF standard 1.01,
   resolution (DPI), density 72x72, segment length 16,
    baseline, precision 8, 256x256, components 3
```

Vamos analisar agora dois exemplos em que iremos determinar o tipo de arquivos texto.

Listagem 4.39: Determinando o tipo de um arquivo `.txt`

```
$ wget -q http://www.gutenberg.org/files/4300/4300-0.
   txt
$ file 4300-0.txt
4300-0.txt: UTF-8 Unicode (with BOM) text, with CRLF
   line terminators
$ file CETENFOLHA
CETENFOLHA: ISO-8859 text, with very long lines
```

Usando o comando **file**, podemos verificar que, apesar de ambos serem arquivos texto, eles utilizam codificações diferentes (cada caractere é representado com uma sequência binária distinta, sendo que esta representação pode ser diferente para diferentes codificações). O padrão ISO-8859 é um padrão de 1987, usualmente utilizado pela Microsoft. O padrão UFT-8 é um padrão mais recente, capaz de representar todos os símbolos do Unicode. Com o advento da Internet, o UFT-8 passou a ser o mais utilizado, entretanto ainda é comum nos depararmos com arquivos texto que utilizam outras formas de codificação.

É importante ressaltar que não existe informação de cabeçalho, ou qualquer outra marcação para indicar a codificação de um arquivo texto. O que o comando **file** faz é um 'chute educado', ou seja, analisando o arquivo ele tenta adivinhar qual é a codificação mais provável.

Para converter entre diferentes forma de codificação, utilizamos o comando **iconv**, tópico da próxima secção.

4.15 Comando iconv

O comando **iconv** é utilizado para realizar conversões de codificação de caracteres, isto é, pegar uma string, texto ou arquivo codificado em um padrão de caracteres e colocá-lo em outro padrão. Este comando pode ser útil, por exemplo, quando temos um arquivo texto feito no Windows e desejamos utilizá-lo no Linux. A Listagem 4.40 ilustra o examplo do comando **iconv**.

Listagem 4.40: Convertendo Padrões de Caracteres

```
$ iconv -f iso-8859-1 -t utf-8 cidades.txt >
   cidadeutf8.txt
```

Em algumas situações a codificação alvo que iremos adotar pode não possuir representação para um determinado caractere. Por exemplo, a codificação ASCII não suporta caracteres acentuados. Se tentarmos realizar a conversão de um texto codificado em UTF-8 com caracteres acentuados para ASCII, obteremos erro. Para realizar a conversão devemos utilizar a opção TRANSLIT, para que se busque converter para o caractere mais próximo (visualmente). Veja o exemplo na Listagem 4.41.

Listagem 4.41: Convertendo Padrões de Caracteres

```
$ echo -e "ação" | iconv -t ASCII -f UTF-8
aiconv: illegal input sequence at position 1
$ echo -e "ação" | iconv -t ASCII//TRANSLIT  -f UTF-8
acao
```

4.16 Comando look

O comando **look** é utilizado para visualizar linhas que possuem uma determinada string. Na Listagem 4.42, vamos procurar no arquivo *cidades.txt* as linhas que possuem a string Bom.

Listagem 4.42: Comando look

```
$ look Belo cidades.txt
Belo Campo (BA)
Belo Horizonte (MG)
Belo Jardim (PE)
Belo Monte (AL)
Belo Oriente (MG)
Belo Vale (MG)
```

4.17 Comando more

O comando **more** é utilizado para processar e visualizar arquivos longos que não cabem na tela do terminal. Ele pode ser utilizado em conjunto com outros comandos. O arquivo *cidades.txt* é um arquivo muito grande e uma opção de visualização é usar o **more** como na Listagem 4.43. Ele coloca as informações suficientes para ocupar o tamanho da tela. Para navegar no arquivo, basta apertar espaço ou o número da página após os dois pontos. Para sair digite q.

Listagem 4.43: Visualizando Arquivos Longos

```
Abadia de Goias (GO)
Abadia dos Dourados (MG)
Abadiânia (GO)
Abaeté (MG)
Abaetetuba (PA)
Abaiara (CE)
Abaíra (BA)
Abaré (BA)
Abatiá (PR)
Abdon Batista (SC)
Abelardo Luz (SC)
Abel Figueiredo (PA)
Abre-Campo (MG)
Abreu e Lima (PE)
Abreulândia (TO)
Acaiaca (MG)
Açailândia (MA)
Acajutiba (BA)
Acará (PA)
Acarape (CE)
Acaraú (CE)
Acari (RN)
Acauã (PI)
:
```

4.18 Comando nl

O comando **nl** enumera as linhas de um arquivo. A Listagem 4.44 apresenta o uso do comando processando o arquivo *cidades.txt*.

Listagem 4.44: Contando o Número de Linhas

```
$ nl  cidades.txt
```

```
...
  5570  Xexeu (PE)
  5571  Xinguara (PA)
  5572  Xique-Xique (BA)
  5573  Zabele(PB)
  5574  Zacarias (SP)
```

4.19 Comando paste

O comando **paste** é interessante para unir linhas de arquivos diferentes formatando as colunas. Imagine que eu tenha separado em dois arquivos, *nomes.txt* com os nomes dos alunos e *notas.txt* com as notas dos alunos. A Listagem 4.45 mostra os dois arquivos.

Listagem 4.45: Arquivos de Exemplo para o Comando paste

```
$ cat nomes.txt
alessandro
pedro
marcos
andre
luciana

$ cat notas.txt
10 20
30 30
40 10
50 60
NC 10
```

Vamos supor que agora eu necessite juntar os dois arquivos em colunas como na Listagem 4.46.

Listagem 4.46: Combinando Dois Arquivos com o Comando paste

```
$ paste nomes.txt notas.txt
alessandro 10 20
pedro 30 30
marcos 40 10
andre 50 60
luciana NC 10
```

Se quiser juntar os arquivos e criar um arquivo CSV (Comma-separated values), poderá utilizar o parâmetro **-d** e especificar que o separador será a vírgula.

Listagem 4.47: Combinando arquivos para formar um arquivo CSV

```
$ paste -d , <(seq 1 3) <(seq 4 6) <(seq 7 9)
1,4,7
2,5,8
3,6,9
```

4.20 Comando rev

O comando **rev** inverte os caracteres de uma linha e envia o resultado para a saída padrão. É bastante útil para a manipulação de caracteres e programas de codificação. Ele recebe como argumento um arquivo ou uma mensagem. A Listagem 4.48 mostra a sintaxe para passar um arquivo como argumento,

Listagem 4.48: Comando para Inverter os Caracteres - rev

```
$ cat  arquivo |  rev
```

A Listagem 4.49 apresenta a sintaxe para passar uma frase como argumento.

Listagem 4.49: Invertendo Caracteres com rev

```
$ echo 'teste' | rev
etset
```

4.21 Comando sort

Para ordenar arquivos utilizamos o comando **sort** . Seja um arquivo *teste.txt* com os seguintes itens na Listagem 4.50.

Listagem 4.50: Ordenando Arquivos com sort

```
abacate
maca
pera
uva
graviola
limao
mamao
```

Para ordenar o arquivo *teste.txt* vamos utilizar o comando **sort** na Listagem 4.51.

Listagem 4.51: Ordenando Arquivos com sort

```
$ cat teste.txt | sort -n
```

```
abacate
graviola
limao
maca
mamao
pera
uva
```

4.22 Comando uniq

O comando **uniq** é usado para encontrar linhas únicas num arquivos, i.e., ele remove linhas duplicadas consecutivas contidas em arquivos. É importante que o arquivo já esteja organizado para que ele possa remover todas as duplicações. Geralmente esse comando trabalha em conjunto com o sort. Ele possui as seguintes opções :

- -c : conta quantas vezes cada linha apareceu
- -u: imprime somente as linhas únicas
- -d: imprime somente linhas duplicadas

Vamos supor que você quer saber quantas palavras distintas existem em uma lista de palavras. Vamos utilizar a combinação dos seguintes comandos:

- sort : ordena as palavras
- uniq: retira frases com palavras iguais
- wc: conta as palavras

A Listagem 4.52 ilustra o comando

Listagem 4.52: Comando uniq

```
$ sort /usr/share/dict/words | uniq | wc -w
99171
```

Caso deseje contar as palavras em um texto, por exemplo, em Dom Casmurro de Machado de Assis. Vamos utilizar os comandos vistos anteriormente para realizar esta tarefa. Primeiramente vamos substituir todas maiúsculas por minúsculas, em seguida vamos remover todos os caracteres que não estiverem entre `a-z` e também não forem espaço (espaço em branco, tabulação, quebra de linha). Feito isso, iremos substituir todo espaço por quebra de linha, em seguida ordenar as palavras, contabilizar apenas uma ocorrência de cada palavra e, por fim, contar quantas linhas foram geradas, ou seja, quantas são as palavras (tipos) utilizadas no texto. O código utilizado é ilustrado na Listagem 4.53.

Listagem 4.53: Contando quantas palavras distintas existem em um texto

```
$ cat dom_casmurro.txt | tr 'A-Z' 'a-z' | tr -dc 'a-z
   [:space:]' | tr [:space:] '\n' | sort | uniq | wc -
   l
9125
```

Agora vamos fazer algo um pouco mais avançado. O início do script é bem similar ao anterior. Incluímos a substituições de caracteres do tipo espaço (espaço, tabulação, linha) por um quebra de linha. Antes de ordenar o resultados vamos utilizar o **sed** para remover as linhas em branco. Depois ordenamos e ao usar o **unique** para obter as ocorrências únicas, vamos solicitar que ele faça também a contagem de quantas vezes cada palavra apareceu, usando para tanto o parâmetro **-c**. Depois vamos ordenar novamente, mas desta vez de forma numérica, usando o parâmetro **-n** e reversa (descendente), usando o parâmetro **-r**. Por fim, vamos adicionar o número das linhas e usar o resultado para fazer um gráfico com o **gnuplot**.

Listagem 4.54: Gráfico da frequência de ocorrência das palavras de um texto

```
$ cat dom_casmurro.txt | tr 'A-Z[:space:]' 'a-z\n' |
   tr -dc 'a-z[:space:]' | sed '/^[[:space:]]*$/d' |
   sort | uniq -c | sort -rn | nl | gnuplot -e "set
   terminal pdf; set output 'palavras.pdf'; set title
   'frequencia das palavras em Dom Casmurro'; set
   ylabel 'frequencia'; set xlabel 'ranque'; set
   logscale xy; plot '-' using 1:2 with lines title '
   dom casmurro'"
```

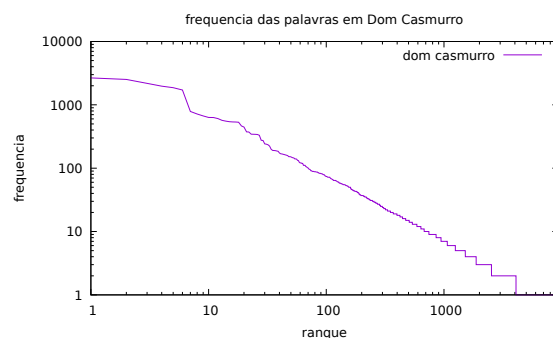



Figura 4.1: Frequência de Ocorrência das palavras em Dom Casmurro

4.23 Comando wc

O comando **wc** conta o número de linhas, palavras e bytes de um arquivo. É possível também contar o número de caracteres. Note que, nem sempre um caractere corresponde a um byte. Isto vai variar com o tipo de codificação utilizada. No ASCII, cada byte corresponde a um caractere, mas no UTF-8 os caracteres são representados com um número variável de bytes (de 1 a 4 bytes). A Listagem 4.55 apresenta o uso do comando processando o arquivo *cidades.txt*.

Listagem 4.55: Contando o Número de Linhas, Palavras e Bytes

```
$ wc cidades.txt
    5593   15850  101042 cidades.txt
```

O comando imprime o número de linhas (5593), palavras (15850) e bytes (101042). Para contar apenas o número de linhas utilize o comando **wc -l** como na Listagem 4.56.

Listagem 4.56: Contando o Número de Linhas

```
$ wc -l cidades.txt
    5593
```

Para contar o número de palavras utilize o comando **wc -w** como na Listagem 4.57.

Listagem 4.57: Contando o Número de Palavras

```
$ wc -w cidades.txt
   15850 cidades.txt
```

Para contar o número de bytes utilize o comando **wc -c** como na Listagem 4.58, ou utilize, **wc -m** para contar o número de caracteres.

Listagem 4.58: Contando o Número de Bytes

```
$ wc -c cidades.txt
  101042 cidades.txt
```

Veja que o número de bytes e caracteres é diferente quando utilizamos caracteres acentuados.

Listagem 4.59: Contando o Número de Bytes e Caracteres

```
$ echo "áéíóú" | wc -c
11
$ echo "áéíóú" | wc -m
6
```

Capítulo 5

Comandos de Sistema

Sumário

5.1 Gerando Todos os Comandos

Se você deseja saber todos os comandos incluídos em sua distribuição, basta digitar o comando **compgen** . Para isto, utilize a opção `-c` como na Listagem 5.1.

Listagem 5.1: Opções Múltiplas

```
$ compgen -c
if
then
else
elif
fi
...
xdvi-xaw
xdvipdfmx
xelatex
xetex
xindy
xindy.run
xmltex
```

5.2 Quem sou eu e onde estou?

Se você tem problemas de múltiplas personalidades, esta é uma boa opção. Brincadeiras a parte, estes comandos são muito importantes no uso diário.

- Quem sou eu? Para saber quem é você (seu username) utilize o comando **whoami** . Este comando é muito utilizado para saber com que usuário você está logado. Às vezes fica muito confuso quando somos o superusuário, ou quando utilizamos mais de um usuário no sistema.
- Onde estou? Para saber o local em que você se encontra na árvore de diretórios (diretório de trabalho ou diretório corrente) use o comando **pwd** .

A Listagem 5.2 apresenta o resultado dos comandos.

Listagem 5.2: Comando whoami e pwd

```
$ whoami
pedro
$ pwd
/home/pedro
```

5.2.1 Arquivo passwd

Quando cadastramos um usuário no sistema Linux, é criado uma entrada no arquivo */etc/passwd*. Neste arquivo ficam armazenados todos os logins e algumas informações sobre os usuários do Linux. Quer listar o conteúdo do */etc/passwd* ? Basta utilizar o comando **cat** . A Listagem 5.3 apresenta o resultado do comando.

Listagem 5.3: Arquivo passwd

```
 $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
...
```

Note que cada linha do arquivo */etc/passwd* contém 7 campos separados por dois-pontos (:) para fornecer algumas informações sobre os usuários do sistema. Em ordem, os campos são

1. usuário;
2. senha (o caractere `x` é utilizado para informar que a senha está encriptada, neste caso, armazenada em */etc/shadow*);
3. id do usuário (zero é reservado para o administrador e os números de 1 a 99 são reservados para algumas contas pré-definidas);
4. id do grupo primários (veja os grupos no arquivo */etc/group*);
5. informação sobre o usuário;
6. localização do diretório *home* do usuário;
7. caminho absoluto para um comando ou shell (*/bin/bash*)

5.3 Comando id

Quando se cria um usuário, **login**, tanto no Linux quanto no Mac, ele recebe um identificador numérico (inteiro) indicando o número do usuário no sistema. É como se fosse o CPF do usuário no sistema e qualquer manipulação será realizada em cima do número do usuário e não do nome. Como todos sabem, computadores são melhores em manipulação de números, índices, do que realizar operações em nomes.

O comando **id** faz a relação entre usuários e identificadores de usuários. A Listagem 5.4 e 5.5 apresenta o resultado do comando para Linux e Mac

respectivamente. Como pode ser observado, meu login no Mac tem identificador 501 e no Linux 1000.

Listagem 5.4: Identificadores no Mac

```
$ id
uid=501(alessandrovivas) gid=20(staff) groups=20(staff
    ),402(com.apple.sharepoint.group.1),12(everyone)
    ,61(localaccounts),79(_appserverusr),80(admin),81(
    _appserveradm),98(_lpadmin),33(_appstore),100(
    _lpoperator),204(_developer),398(com.apple.
    access_screensharing),399(com.apple.access_ssh)uid
    =501(alessandrovivas) gid=20(staff) groups=20(staff
    ),402(com.apple.sharepoint.group.1),12(everyone)
    ,61(localaccounts),79(_appserverusr),80(admin),81(
    _appserveradm),98(_lpadmin),33(_appstore),100(
    _lpoperator),204(_developer),398(com.apple.
    access_screensharing),399(com.apple.access_ssh)
```

Listagem 5.5: Identificadores no Linux

```
uid=1000(vivas) gid=1000(vivas) grupos=1000(vivas),4(
    adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(
    lpadmin),124(sambashare)
```

5.4 Alterando a Senha

O comando **passwd** permite a alteração da senha pelo usuário a qualquer tempo. A Listagem 5.6 apresenta o procedimento para alteração da senha.

Listagem 5.6: Alterando a Senha

```
$ passwd
Changing password for maria
# digite a sua senha atual
(current) UNIX password:
# entre com a sua nova senha
Enter new UNIX password:
# digite novamente sua senha
Retype new UNIX password:
passwd: password updated successfully
```

Será necessário fornecer a senha atual para que o sistema autorize a mudança. A senha deve ser digitada e re-digitada para que a mudança seja

efetuada. Em sistemas configurados com sistemas de autenticação centralizada como LDAP e NIS, este comando pode ser utilizado, e a senha será alterada no servidor de autenticação.

5.4.1 Usuários Logados

Quer descobrir quais são os usuários que estão logados nos sistema? Utilize o comando **users** da Listagem 5.7.

Listagem 5.7: Usuários Logados

```
$ users
vivas vivas
```

5.5 Comando finger

O comando **finger** fornece informações sobre os usuários cadastradas no sistema. Dentre essas informações estão: nome, login, diretório inicial, último login efetuado com sucesso, shell de uso. Ao se executar o comando **finger** sem argumentos, ele exibe num formato padrão de informação. Utilize o argumento `-l` para listar informações de todos os usuários logados na máquina naquele momento. A Listagem 5.9 apresenta o resultado para o sistema Mac, mas é similar ao Linux. Caso o comando não esteja instalado, proceda a instalação do mesmo, Listagem 5.8.

Listagem 5.8: Instalação do finger

```
$ sudo apt-get install finger
Password:
```

Para visualizar informações sobre um usuário específico, utilize a Listagem 5.9.

Listagem 5.9: Comando finger

```
$ finger vivas
Login: vivas                                    Name: vivas
Directory: /home/vivas                          Shell: /bin/
   bash
On since Wed Sep  4 15:00 (-03) on pts/1 from
   170.83.103.236
   12 hours 22 minutes idle
On since Thu Sep  5 20:18 (-03) on pts/2 from
   10.2.212.106
   2 seconds idle
No mail.
No Plan.
```

Para apresentar todos os usuários que estão logados naquele momento é usado **finger -l**. A Listagem 5.10 do resultado no mac.

Listagem 5.10: Comando finger -l

```
$ finger -l vivas
Login: vivas                              Name: vivas
Directory: /home/vivas                    Shell: /bin/
   bash
On since Wed Sep  4 15:00 (-03) on pts/1 from
   170.83.103.236
   12 hours 22 minutes idle
On since Thu Sep  5 20:18 (-03) on pts/2 from
   10.2.212.106
   3 seconds idle
No mail.
No Plan.
```

Outra variação é utilizar o comando sem argumentos, isto é, digitando apenas **finger**. A coluna `Login` é o nome do login do usuário, a coluna `Name` é o nome completo do usuário, `Tty` é o terminal onde o usuário está logado, `Idle` mostra o tempo ocioso, `Login Time` mostra a data e a hora quando o usuário logou, `Office` mostra a localização física do usuário e `Office Phone` mostra o telefone do usuário. A Listagem 5.11 mostra o resultao do comando no Linux

Listagem 5.11: Comando finger no Linux

```
$ finger
Login     Name       Tty      Idle  Login Time
   Office     Office Phone
leoca     leoca     *:0             Oct 29 11:51 (:0)
leoca     leoca      pts/0       1  Oct 29 11:52 (:0)
leoca     leoca      pts/1      25  Oct 29 11:52 (:0)
leoca     leoca      pts/2       1  Oct 29 12:13 (:0)
leoca     leoca      pts/3          Oct 29 12:17 (:0)
```

5.6 Comando free

O comando **free** mostra a estatística de uso de memória, incluindo memória livre total, memória utilizada, memória física, memória swap, memória compartilhada e buffers utilizados pelo kernel. A Listagem 5.13 apresenta o resultado deste comando para o Sistema Operacional Linux. No Mac pode-se utilizar o comando **vm_stat** apresentado na Listagem 5.12.

Listagem 5.12: Comando free

```
Mach Virtual Memory Statistics: (page size of 4096
   bytes)
Pages free:                                    2151369.
Pages active:                                   796598.
Pages inactive:                                 127372.
Pages speculative:                              411792.
Pages throttled:                                     0.
Pages wired down:                               706471.
Pages purgeable:                                 72490.
"Translation faults":                          3674787.
Pages copy-on-write:                            147901.
Pages zero filled:                             2462641.
Pages reactivated:                                  10.
Pages purged:                                        0.
File-backed pages:                              620935.
Anonymous pages:                                714827.
Pages stored in compressor:                          0.
Pages occupied by compressor:                        0.
Decompressions:                                      0.
Compressions:                                        0.
Pageins:                                        151389.
Pageouts:                                            0.
Swapins:                                             0.
Swapouts:                                            0.
```

Listagem 5.13: Comando free

```
$ free
              total        used        free
                 shared  buff/cache   available
Mem:      132008792     1137996    99224552
   2672    31646244   129798924
Swap:       8388604           0     8388604
```

5.7 Comando su

Executa o interpretador de comandos com a substituição do usuário e do grupo. Possibilidade de logar imediatamente no mesmo terminal em uso com outro usuário. Prática comum de super-usuário. A Listagem 5.14 apresenta a execução do comando **su** .

Listagem 5.14: Logar como Super Usuário

```
$ su -
Password:
```

5.8 Comando uname

O comando **uname** é utilizado para apresentar informações sobre o sistema operacional de sua máquina. A Listagem 5.15 apresenta o comando, no Linux, para verificar qual sistema operacional está utilizando. A Listagem 5.16 ilustra o resultado para o Mac.

Listagem 5.15: Verificar Informações sobre o Linux

```
 uname -s
Linux
```

Listagem 5.16: Exemplo do Sistema Operacional Mac

```
$ uname -s
Darwin
```

Para verificar a versão do seu kenel utilize o comando apresentando na Listagem 5.17.

Listagem 5.17: Verificar sua Versão do kernel

```
$ uname -r
4.9.0-9-amd64
```

A Listagem 5.18 apresenta o comando para verificar se sua plataforma é de 32 ou 64 bits. Neste caso, a plataforma é de 64 bits, pois a resposta foi **x86_64**.

Listagem 5.18: Verificando a Plataforma

```
$ uname -m
x86_64
```

Para descobrir o nome de sua máquina, utilize o comando **uname -n**. A Listagem 5.19 apresenta o resultado.

Listagem 5.19: Verificar o Nome de sua Máquina

```
$ uname -n
musashi
```

Para apresentar todas as informações sobre seu sistema operacional, utilize o comando **uname -a**. A Listagem 5.20 apresenta o resultado.

Listagem 5.20: Apresenta todas as informações sobre seu sistema operacional

```
$ uname -a
Linux vivas-VirtualBox 3.8.0-33-generic #48~precise1-
   Ubuntu SMP Thu Oct 24 16:28:06 UTC 2013 x86_64
   x86_64 x86_64 GNU/Linux
```

5.9 Comando uptime

O comando **uptime** apresenta as seguintes informações: a hora corrente, há quanto tempo o seu computador está ligado, quantidade de usuários logados e a carga média do sistema a 1, 5 e 15 minutos passados. A Listagem 5.21 ilustra o resultado do comando.

Listagem 5.21: Tempo de Funcionamento

```
$ uptime
 20:29:23 up 3 days, 11:57,  2 users,  load average:
    0.00, 0.02, 0.00
```

5.10 Verificando a versão de um comando

Os comandos são programas e apresentam versões. Caso queira verificar a versão de um comando, utilize a opção *version* como na Listagem 5.22.

Listagem 5.22: Versão de um comando

```
$ ls --version
ls (GNU coreutils) 8.13
Copyright (C) 2011 Free Software Foundation, Inc.
Licenca GPLv3+:: GNU GPL versao 3 ou posterior <http
   ://gnu.org/licenses/gpl.html>
Este e um software livre: voce e livre para altera-lo
   e redistribui-lo
NAO HA GARANTIA, na maxima extensao permitida pela lei
   .


Escrito por Richard M. Stallman e David MacKenzie.
```

5.11 Variável PATH

Ao se digitar um comando, o arquivo deve ser localizado pelo sistema operacional para ser executado. Se ele não o encontra, uma mensagem de erro é

exibida em seguida. Algumas vezes, quando criamos, por exemplo, arquivos executáveis, necessitamos passar o local do arquivo como por exemplo:

Listagem 5.23: Localização de um Comando

```
$./arquivo_executavel
$./usr/bin/arquivo_executavel
```

Acima, o ponto indica o caminho desde o diretório raiz até o diretório corrente. Entretanto, existem locais padrões a serem buscados, e tais locais são definidos por uma variável chamada PATH.

Quando você digita um comando e o shell não o encontra, pode estar acontecendo duas coisas: o comando não foi instalado ou o seu shell não está procurando no local correto. Para saber todos os caminhos onde seu shell procura os comandos, digite:

Listagem 5.24: Variável PATH

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
```

Os caminhos são separados por dois-pontos (:). No exemplo acima, o primeiro local onde o shell procura os comandos é o diretório */usr/local/bin*. Como pode ser observado, o shell não procura diretamente em seu diretório. Você pode imaginar o motivo? Razões de segurança: se um intruso inserisse um programa malicioso (ex. uma modificação do comando ls) e se o shell procurasse diretamente no seu diretório, a execução desse programa poderia danificar seus arquivos.

5.12 Comando timeout

O comando **timeout** executa um comando com limite de tempo. A Listagem 5.25 apresenta o uso do comando.

Listagem 5.25: Comando timout

```
$ timeout 5 ping www.ufvjm.edu.br
PING www.ufvjm.edu.br (200.131.252.165) 56(84) bytes
   of data.
64 bytes from 200.131.252.165 (200.131.252.165):
   icmp_seq=1 ttl=58 time=0.619 ms
64 bytes from 200.131.252.165 (200.131.252.165):
   icmp_seq=2 ttl=58 time=0.638 ms
64 bytes from 200.131.252.165 (200.131.252.165):
   icmp_seq=3 ttl=58 time=0.655 ms
```

```
64 bytes from 200.131.252.165 (200.131.252.165):
   icmp_seq=4 ttl=58 time=0.563 ms
64 bytes from 200.131.252.165 (200.131.252.165):
   icmp_seq=5 ttl=58 time=0.642 ms
```

5.13 Comando w

O comando **w** verifica quais usuários estão logados e o que eles estão fazendo. A Listagem 5.26 apresenta o resultado do comando.

Listagem 5.26: Comando w

```
$ w
 7:44  up 10 days, 16:57, 3 users, load averages: 2,21
     2,01 2,26
USER     TTY      FROM              LOGIN@  IDLE WHAT
alessandrovivas console  -                24Ago15 10
   days -
alessandrovivas s000     -                24Ago15
   15:47 -bash
alessandrovivas s001     -                26Ago15
   - w
```

5.14 Comando whereis

O comando **whereis** determina a localização de seu programa executável (binário), fonte e páginas de manual referente a um comando. Ele é mais completo que o comando **which**.

Caso deseje encontrar a localização de um programa, por exemplo o **gcc**, utilizamos o exemplo na Listagem 5.27.

Listagem 5.27: Comando whereis

```
$ whereis gcc
gcc: /usr/bin/gcc /usr/lib/gcc /usr/bin/X11/gcc /usr/
   share/man/man1/gcc.1.gz
```

5.15 Comando locate

O comando **locate** lista arquivos que contenham o texto dado (semelhante ao whereis). A Listagem 5.28 utiliza o comando para listar todas as ocorrências de passwd.

Listagem 5.28: Comando locate

```
$ locate passwd
/etc/passwd
/etc/passwd-
/etc/cron.daily/passwd
/etc/init/passwd.conf
/etc/init.d/passwd
/etc/pam.d/chpasswd
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd
/usr/bin/grub-mkpasswd-pbkdf2
/usr/bin/lppasswd
/usr/bin/mkpasswd
/usr/bin/passwd
/usr/bin/smbpasswd
/usr/bin/vino-passwd
...
```

5.16 Comando which

Quer descobrir se um programa está no seu PATH? Utilize o comando **which** para resolver este problema como ilustrado na Listagem 5.29.

Listagem 5.29: Comando which

```
$ which ls
/bin/ls
$ which gcc
/usr/bin/gcc
```

5.17 Comando whatis

O comando **whatis** pode ser utilizado para buscar informações de comandos no banco de dados do sistema, como na Listagem 5.30.

Listagem 5.30: Comando whatis

```
vivas@zafu:~$ whatis clear
clear (1)            - clear the terminal screen
vivas@zafu:~$ whatis ps
ps (1)               - report a snapshot of the
   current processes.
```

```
vivas@zafu:~$ whatis netstat
netstat (8)          - Mostra conexoes de rede,
   tabelas de roteamento, estatisticas de interface e
   conexoes...
vivas@zafu:~$ whatis route
route (8)            - mostra / manipula a tabela de
   roteamento IP
vivas@zafu:~$ whatis ls
ls (1)               - list directory contents
LS (6)               - display animations aimed to
   correct users who accidentally enter LS instead of
   ls .
vivas@zafu:~$ whatis touch
touch (1)            - change file timestamps
vivas@zafu:~$ whatis cat
cat (1)              - concatenate files and print on
   the standard output
vivas@zafu:~$ whatis date
date (1)             - print or set the system date
   and time
```

5.18 Comando who

O comando **who** pode ser utilizado para verificar quem está logado no computador. A Listagem 5.31 apresenta o resultado do comando e mostra que estou logado no computador através de dois endereços IP.

Listagem 5.31: Comando who

```
vivas@musashi:~$ who
vivas    pts/1        2019-09-04 15:00 (171.95.2.236)
vivas    pts/2        2019-09-05 20:18 (10.2.212.106)
```

5.19 Executando múltiplos comandos

Para executar múltiplos comandos, basta separá-los com ponto e vírgula. A Listagem 5.32 apresenta o resultado dos comandos.

Listagem 5.32: Rodando Múltiplos Comandos

```
$ hostname; date; ls;
zafu
Sex Jan  3 17:22:40 BRST 2014
```

```
Area de Trabalho  Docs       Downloads
   ifconfig1-modificado  Musica   teste2.txt  ubuntu
arq1-hard         Docs2      examples.desktop
   Imagens               Publico  teste3.txt  Videos
arq2              Documentos  ifconfig1
   Modelos               teste    teste.txt
```

O exemplo ilustrado na Listagem 5.32 mostra como rodar comandos em sequência, de forma que eles sempre serão executados, independente do resultado da execução do comando anterior. Uma outra forma é condicionar a execução do comando subsequente à execução do primeiro comando. Podemos utilizar o `&&` para executar o segundo comando apenas se o primeiro comando tiver sucesso. Outra opção é condicionar a execução do segundo ao fracasso do primeiro comando, ou seja, o segundo só será executado se o primeiro retornar erro. Para este fim utilizaremos o `||`. Ambos exemplos são ilustrados na Listagem 5.33.

Listagem 5.33: Rodando Múltiplos Comandos Condicionados

```
$ test -f teste.existe && echo 'existe'
$ test -f teste.existe || echo 'existe'
$ touch teste.existe && test -f teste.existe && echo '
   existe'
```

Note que o primeiro só irá imprimir 'existe' na tela se o arquivo *teste.existe* já existir no diretório corrente. Caso o arquivo não exista, apenas a terceira linha imprimirá 'existe' na tela, umas vez que o arquivo acaba de ser criado pelo comando **touch**.

5.20 Executando um comando em background

Outra forma muito útil de executar programas é executá-los em background. Isto implica que o programa estará rodando, mas você não verá os resultados e não poderá passar sinais ao programa por meio do teclado. Isto implicará que o programa não poderá ser finalizado utilizando a combinação de teclas <Ctrl-C>. Ao executar um programa em background, o shell ficará livre para a execução de outros programas. Para executar um programa em background, você deverá utilizar o `&` após chamar o programa. Veja o exemplo na Listagem 5.34, no qual o **emacs** será executado em background.

Listagem 5.34: Rodando Comando em Background

```
$ emacs &
```

Capítulo 6

Gerenciamento de Processos

Sumário

6.1 Visualizando Todos os Processos em Execução

Definimos processo como um programa em execução. Podemos definir como espaço de endereçamento de um processo como: código do programa, variáveis utilizadas, pilha do processo e outras informações necessárias.

Os processos são constituídos da localização do espaço de endereçamento, status, prioridade de execução, informações sobre os recursos utilizados, máscara de sinal do processo e identificação do proprietário. Os principais atributos são: identificador do processo (PID), Identificador do processo pai (PPID), prioridade de execução (nice), TTY (terminal), identificação real do usuário e do grupo.

O comando **ps** com a opção `-aef` apresenta todos os processos em execução como na Listagem 6.1.

Listagem 6.1: Todos os Processos em Execução

```
$ ps -aef
  UID   PID  PPID   C STIME   TTY           TIME CMD
    0     1     0   0 Seg02   ??        11:25.68 /sbin
       /launchd
    0    45     1   0 Seg02   ??         1:36.14 /usr/
       sbin/syslogd
    0    46     1   0 Seg02   ??         0:48.74 /usr/
       libexec/UserEventAgent (System)
    0    48     1   0 Seg02   ??         0:04.10 /usr/
       libexec/kextd
...
```

6.2 Todos os Processos de um Usuário Específico

O comando **ps** com a opção `-u` pode ser utilizado para visualizar todos os processos de um determinado usuário como na Listagem 6.2.

Listagem 6.2: Todos os Processos em Execução de um Usuário Específico

```
$ ps -u alessandrovivas
  UID   PID TTY           TIME CMD
  501   287 ??         0:15.25 /usr/libexec/
     UserEventAgent (Aqua)
  501   289 ??         3:17.49 /usr/sbin/distnoted
     agent
  501   290 ??         0:06.69 /usr/sbin/
     universalaccessd launchd -s
  501   291 ??         0:14.95 /usr/sbin/cfprefsd
     agent
```

```
  501   302 ??         0:08.25 /usr/sbin/usernoted
  501   303 ??         0:04.27 /usr/libexec/
    nsurlsessiond
```

6.3 Lista de Processos Ordenadas pelo Consumo de CPU

Para obter uma lista de processos ordenadas pelo consumo de CPU utilizamos a Listagem 6.3. Neste caso utilizamos o comando **head** para imprimir somente as 5 primeiro ocorrências.

Listagem 6.3: Processos que Mais Consomem CPU

```
$ ps -aef -r | head -5
  PID TTY      STAT   TIME COMMAND
18601 pts/1    D+     0:00 [bash]
18600 pts/1    R+     0:00 ps -aef -r SHELL=/bin/bash
```

6.4 Lista dos Processos que mais Consomem Memória

Para obter uma lista de processos ordenadas pelo consumo de memória utilizamos a Listagem 6.4. Neste caso, utilizamos o comando **head** para imprimir somente as 5 primeiro ocorrências.

Listagem 6.4: Processos que Mais Consomem CPU

```
$ ps -aef -m | head -5
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 ago27 ?        00:00:32 /lib/
   systemd/systemd --system --deserialize 34
root         -     -  0 ago27 -        00:00:32 -
root         2     0  0 ago27 ?        00:00:00 [
   kthreadd]
root         -     -  0 ago27 -        00:00:00 -
```

6.5 Obtendo Informações de um Processo Específico

Para visualizar informações de um processo específico, utilizamos o comando **ps** em conjunto com o comando **grep**, como na Listagem 6.5.

Listagem 6.5: Obtendo Informações de um Processo Específico

```
$ ps -aef | grep firefox
vivas     2934  1893  0 ago27 tty2     00:49:28 /usr/
   lib/firefox-esr/firefox-esr
vivas     3904  2934  0 ago27 tty2     00:47:48 /usr/
   lib/firefox-esr/firefox-esr
vivas     4027  2934  0 ago27 tty2     00:25:16 /usr/
   lib/firefox-esr/firefox-esr
vivas     4207  2934  0 ago27 tty2     00:18:58 /usr/
   lib/firefox-esr/firefox-esr
vivas    12150  2934  0 set03 tty2     00:20:08 /usr/
   lib/firefox-esr/firefox-esr
```

6.6 Comando pstree

O comando **pstree** apresenta todos os comandos em execução no formato de uma árvore relacionando a dependência entre eles. A Listagem 6.6 apresenta o resultado do comando. Neste exemplo, utilizamos o comando **head** apenas para limitar o número de linhas.

Listagem 6.6: Obtendo a Lista de Processos em forma de Árvore

```
$ pstree
systemd   ModemManager -2*[{ModemManager}]
          NetworkManager dhclient
                         2*[{NetworkManager}]
...
```

6.7 Comando top

O comando **top** é utilizado para obter informações sobre os processos que estão rodando em sua máquina. A Figura 6.1 apresenta do resultado do comando.

Os estados possíveis de um processo são:

- runnable - rodando
- sleeping - está esperando por um evento
- swapped - não está executando e foi armazenado na memória virtual
- zombie - está tentando morrer (pode ter perdido seu pai)
- stopped - está proibido de executar (através de CTRL-Z ou um SIGSTOP)

```
vivas@masamune:~/Documentos/Livros/Linux2aEdicao$ top

top - 17:32:05 up 13 days, 53 min,  1 user,  load average: 0,53, 0,88, 1,10
Tasks: 267 total,   2 running, 265 sleeping,   0 stopped,   0 zombie
%Cpu0  :  5,6 us,  1,0 sy,  0,0 ni, 90,5 id,  0,0 wa,  0,0 hi,  2,9 si,  0,0 st
%Cpu1  :  1,0 us,  0,7 sy,  0,0 ni, 97,0 id,  0,3 wa,  0,0 hi,  1,0 si,  0,0 st
%Cpu2  :  2,0 us,  1,0 sy,  0,0 ni, 97,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu3  :  2,0 us,  1,7 sy,  0,0 ni, 96,3 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu4  :  2,7 us,  1,3 sy,  0,0 ni, 96,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu5  :  2,3 us,  1,7 sy,  0,0 ni, 96,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu6  :  4,6 us,  0,7 sy,  0,0 ni, 94,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu7  :  1,3 us,  1,0 sy,  0,0 ni, 97,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7904,7 total,   2323,4 free,   4848,1 used,    733,2 buff/cache
MiB Swap:   8116,0 total,   7958,5 free,    157,5 used.   2600,1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4207 vivas     20   0 2693804 448560  66000 S   8,3   5,5  19:15.18 Web Content
 1893 vivas     20   0 4179188 214188  60520 S   8,0   2,6  29:20.29 gnome-shell
11076 vivas     20   0 4091172 183708  64248 S   4,3   2,3   7:14.41 spotify
 1944 vivas      9 -11 2750200  13664  11060 S   4,0   0,2   7:11.37 pulseaudio
11113 vivas     20   0 1927132 323584 118428 S   2,3   4,0   5:18.58 spotify
12150 vivas     20   0 2836440 826148  91644 S   2,3  10,2  20:17.15 Web Content
 9444 vivas     20   0  420608  40588  26024 S   1,7   0,5   0:17.36 gnome-terminal-
11093 vivas     20   0  989352  42572  21412 S   1,7   0,5   2:47.94 spotify
 3904 vivas     20   0 2814524 677628  90596 S   1,0   8,4  48:39.74 Web Content
 4027 vivas     20   0 3137592 574128  59500 S   1,0   7,1  25:19.87 Web Content
18632 vivas     20   0   11484   3916   3144 R   0,7   0,0   0:00.06 top
    1 root      20   0  170972   9552   6824 S   0,3   0,1   0:32.57 systemd
   10 root      20   0       0      0      0 I   0,3   0,0   0:11.60 rcu_sched
```

Figura 6.1: Comando top

Para sair da tela basta digitar "q".

A sintaxe deste comando é:

Listagem 6.7: Controlando Processos

```
top [-] [d delay] [q] [S] [s] [i]

Onde:
d       Especifica o tempo de atraso entre as atualiza
   \c{c}\~{o}es da tela
q       Força o comando top a atualizar sem atraso
S       Usa o modo cumulativo
s       Roda em modo seguro desabilitando comandos
   interativos
i       Ignora processos zombies
```

Vamos exercitar todos os comandos.

Listagem 6.8: Controlando Processos

```
$ top -d 1
$ top -q
$ top -S
$ top -s
$ top -i
```

6.8 Listando todos os Sinais com o Comando kill

O comando **kill** é utilizado para enviar sinais para um processo. A Listagem 6.9 apresenta todos os sinais disponíveis.

Listagem 6.9: Comando kill

```
$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4)
    SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9)
    SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14)
   SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19)
   SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24)
   SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29)
   SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36)
   SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41)
   SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46)
   SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51)
   SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56)
   SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61)
   SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

6.9 Matando um Processo com o Comando Kill

Para eliminar um processo com o comando **kill** precisamos saber o PID do processo. Vamos supor que você deseja eliminar o processo do firefox. O primeiro passo é saber o PID do Kolourpaint. Para isto, faça o procedimento da Listagem 6.10.

Listagem 6.10: Comando kill

```
$ ps -aef | grep paint
```

```
vivas    12005  1893  0 set03 tty2     00:00:13
   kolourpaint
vivas    18805 15188  0 17:38 pts/1    00:00:00 grep
   paint
$ kill -9 12005
```

Como podemos perceber o PID do firefox é 12005. Agora para matar o processo Kolourpaint enviamos um sinal SIGKILL, número 9, como na Listagem 6.11.

Listagem 6.11: Matando o Processo Firefox

```
$ kill -9 12005
```

6.10 Controlando Processos

No módulo anterior aprendemos que, para finalizar um processo, basta utilizar o comando CTRL-C. Para interromper um processo basta utilizar o comando CTRL-Z. Você sabe o que é um processo?

Processo é um programa que está rodando em seu espaço virtual de endereçamento. Ou melhor, um processo é um programa independente, em estado de execução, que tem seu próprio conjunto de recursos. Tudo que está rodando no Linux é um processo. Uma tarefa, por outro lado, pode envolver vários comandos executando em série.

Podemos definir como espaço de endereçamento: código do programa, variáveis utilizadas, pilha do processo e outras informações necessárias.

Os processos são constituídos da localização do espaço de endereçamento, status, prioridade de execução, informações sobre os recursos utilizados, máscara de sinal do processo e identificação do proprietário. Os principais atributos são: identificador do processo (PID), Identificador do processo pai (PPID), prioridade de execução (nice), TTY (terminal), identificação real do usuário e do grupo.

Os processos podem ser criados utilizando a rotina fork. Com o fork você cria uma cópia idêntica ao processo pai, mas com outro PID.

O sistema operacional linux tem vários tipos de processos:

- Processos interativos: é um processo inicializado (e controlado por) um Shell. Um processo interativo pode estar em background ou foreground.

- Processos batch: é um processo que não está associado a um terminal, mas é submetido a uma fila para ser executado sequencialmente.

- Processos daemon : é um processo que fica rodando em background até ser requisitado. Este tipo de processo é usualmente gerado no processo de inicialização da máquina.

Um programa no Linux pode ser executado de duas formas: primeiro plano (foreground) e segundo plano (background). Ao executar no primeiro plano, devemos esperar o término da execução do comando para entrar com um novo comando. Somente é mostrado o aviso de comando após o término de execução do comando/programa. Isto fica bem claro quando executamos o comando **ls**, somente entramos com um novo comando quando ele termina sua execução; isto é a forma usual de execução de comandos no Linux. Quando trabalhamos com a execução em segundo plano não precisamos esperar o término da execução de um programa para executar um novo comando. O comando fica sendo executado internamente e ao terminar ele mostra uma mensagem de finalização acompanhada do número PID do processo que terminou.

Para colocar um processo rodando em segundo plano colocamos o modificador "&".

Listagem 6.12: Controlando Processos

```
# entre como root
$ find / -name boot.b &
[1] 7998
/boot/boot.b
```

O comando para listar os processos que estão rodando em sua máquina é o **ps** (process status). Este comando pode ser usado por todos os usuários, mas sua saída muda quando você é o root.

Listagem 6.13: Controlando Processos

```
$ ps
  PID TTY          TIME CMD
 7012 pts/0    00:00:00 sh
 7081 pts/0    00:00:00 ps
```

Este comando é organizado em colunas. A primeira coluna, PID, indica o número de identificação do processo. Todos os processos que rodam no Linux recebem um identificador (número inteiro) e para manipulação dos processos devemos utilizar este número. Este número inicia em 0 e é incrementado de 1 para cada novo processo, o número final é 65564. Quando o Linux chega ao último número, ele começa a numeração do menor número pulando os que estiverem ativos. Os processos que possuem menor número são os dos sistemas do kernel e os daemons, que iniciam quando o Linux é inicializado (boot) e continuam ativo enquanto o sistema estiver rodando.

A coluna TTY no comando **ps** mostra em qual terminal você iniciou o processo. A coluna STAT mostra o status corrente do processo, os estados podem ser:

- S processo está dormindo

- R processo está rodando.

Um processo está dormindo quando ele não está ativo. A coluna STAT não apareceu quando rodamos o comando ps. Um processo está rodando quando ele está ativo na CPU.

A coluna TIME mostra a quantidade de tempo da CPU que o processo está utilizando. Deve ser ressaltado que é a quantidade de tempo da CPU e não a quantidade de tempo que o processo está ativo.

A última coluna indica o nome do processo que está rodando. Este nome é usualmente o comando que você digitou. No exemplo acima, temos dois comandos o **sh** (estou acessando a máquina remotamente via "sh") e o **ps** (process status) comando que acabei de digitar.

Outro conceito importante é o parentesco entre processos. Quando um processo inicia um segundo processo, o segundo processo é chamado de processo filho.

Este comando tem várias variações e começaremos a estudá-las agora. O **ps -u** é o comando que lista os processos que estão rodando

Listagem 6.14: Controlando Processos

```
$ ps -u joao
PID TTY          TIME CMD
 7011 ?        00:00:00 sshd
 7012 pts/0    00:00:00 sh
 7295 pts/0    00:00:00 ps
$ ps u
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT
   START   TIME COMMAND
joao      7012  0.0  1.0  2456 1304 pts/0    S    08:46
      0:00 -sh
joao      7221  0.0  1.1  3288 1404 pts/0    R    09:19
      0:00 ps u
```

Para matar um processo, parar ou continuar, o Linux utiliza uma forma especial de comunicação chamada de sinais. É a mesma coisa quando usamos os comandos CTRL-C e CTRL-Z. Para fazer estes testes vamos utilizar o comando **top**.

Para colocar um processo rodando em background, vamos usar o comando com o top.

Listagem 6.15: Controlando Processos

```
$ top &
[1] 7528

[1]+  Stopped(SIGTTOU)        top
```

Para matar, parar ou continuar processos utilizamos os sinais. Para enviar um sinal para um processo utilizamos o comando **kill**. Como colocamos o processo top rodando em background, vamos utilizar o comando **jobs**.

Listagem 6.16: Controlando Processos

```
$ jobs -l
[1]+  7528 Parado (saída tty)       top
```

Como podemos ver, a tarefa top está parada e o pid é o número 7528. Para matar este processo usamos o comando kill mais o número do processo:

Listagem 6.17: Controlando Processos

```
$ kill -9 7528
$ jobs -l
[1]+  7528 Morto                    top
```

Assim o processo está morto e se listarmos novamente nada vai aparecer:

Listagem 6.18: Controlando Processos

```
$ jobs -l
```

Agora vamos colocar o processo top rodando novamente:

Listagem 6.19: Controlando Processos

```
$ top
// entre em outro terminal e liste os processos que
   estão rodando
// use o comando ps -u "nome_do_seu_usuário"
$ ps -u joao
  PID TTY          TIME CMD
 7011 ?        00:00:01 sshd
 7012 pts/0    00:00:00 sh
 7587 pts/0    00:00:00 top
 7596 ?        00:00:00 sshd
 7597 pts/1    00:00:00 sh
 7599 pts/1    00:00:00 ps
// o número do processo top é 7487, agora vamos mandar
    um sinal para
// este processo parar sua execução
$ kill -s SIGSTOP 7587
// entre no outro terminal e liste as tarefas que estã
   o rodando
$ jobs -l
[1]+  7587 Parado (sinal)           top
```

```
// agora vamos enviar um sinal para matá-lo, além do
   valor numério
// podemos utilizar a palavra SIGKILL acompanhada do n
   úmero do
// processo
joao@dcomp:~$ kill -s SIGKILL 7587
joao@dcomp:~$ jobs -l
[1]+  7587 Morto                        top
```

Assim, podemos utilizar o comando **kill** com valores numéricos ou a opção `-s` onde passamos nome do sinal a ser enviado. Outra opção é o sinal SIGHUP que faz com que o processo releia seu arquivo de configuração. O sinal SIGSTOP mantêm o processo parado até ele receber o sinal SIGCONT, vamos testar isto:

Listagem 6.20: Controlando Processos

```
// entre em um terminal e digite:
$ top
// entre em outro terminal e liste os processos que
   estão rodando
// use o comando ps -u "nome do seu usuário"
$ ps -u joao
 PID TTY          TIME CMD
 7011 ?        00:00:01 sshd
 7012 pts/0    00:00:00 sh
 7596 ?        00:00:00 sshd
 7597 pts/1    00:00:00 sh
 7660 pts/0    00:00:00 top
 7662 pts/1    00:00:00 ps
// o número do processo top é 7660, agora vamos mandar
    um sinal para
// este processo parar sua execução
$ kill -s SIGSTOP 7660
// entre no outro terminal e liste as tarefas que estã
   o rodando
$ jobs -l
joao@dcomp:~$ jobs -l
[1]+  7660 Parado (sinal)          top
// agora vamos enviar um sinal para continuar a sua
   execução, além do // valor numérico podemos
   utilizar a palavra SIGCONT acompanhada do // número
    do processo
joao@dcomp:~$ kill -s SIGCONT 7660
```

Se o comando funcionou normalmente, no outro terminal o **top** continuou sua execução. Agora termine o mesmo digitando "q".

Para finalizar um processo pelo nome utilizamos o comando **killall**. Tente descobrir qual o terminal você está "logado"e mande um sinal para matá-lo.

Listagem 6.21: Controlando Processos

```
$ ps
$ killall -9 "nome_do_processo"
```

Vamos agora utilizar outras opções do comando **ps**. Deve ser ressaltado que o hífen neste comando não é necessário.

Tente utilizar ao opção **ps -aux**:

Listagem 6.22: Controlando Processos

```
$ ps aux
root@dcomp:/usr/lib# ps aux
USER        PID %CPU %MEM   VSZ  RSS TTY       STAT
   START   TIME COMMAND
root          1  0.0  0.3  1272  444 ?         S
   Dec11   0:00 init
root          2  0.0  0.0     0    0 ?         SW
   Dec11   0:00 [keventd]
root          3  0.0  0.0     0    0 ?         SWN
   Dec11   0:00 [ksoftirqd_CPU0]
root          4  0.0  0.0     0    0 ?         SW
   Dec11   0:05 [kswapd]
root          5  0.0  0.0     0    0 ?         SW
   Dec11   0:00 [bdflush]
root          6  0.0  0.0     0    0 ?         SW
   Dec11   0:01 [kupdated]
root        133  0.0  0.4  1468  576 ?         S
   Dec11   0:00 /sbin/dhclient-2.2.x -q eth0
root        208  0.0  0.4  1344  596 ?         S
   Dec11   0:03 /sbin/syslogd
root        211  0.0  0.8  1836 1044 ?         S
   Dec11   0:01 /sbin/klogd
root        243  0.0  0.7  2772  960 ?         S
   Dec11   0:10 /usr/sbin/nmbd -D
root        245  0.0  0.6  3556  788 ?         S
   Dec11   0:00 /usr/sbin/smbd -D
root        342  0.0  0.8  2788 1052 ?         S
   Dec11   0:00 /usr/sbin/sshd
root        352  0.0  0.6  2056  832 ?         S
   Dec11   0:00 /usr/sbin/xinetd -reuse
```

```
nobody      355  0.0  0.8  2936 1028 ?        S
   Dec11   0:00 proftpd (accepting connections)
daemon      358  0.0  0.4  1384  556 ?        S
   Dec11   0:00 /usr/sbin/atd
root        361  0.0  0.5  1652  680 ?        S
   Dec11   0:00 /usr/sbin/cron
root        365  0.0  1.0  2932 1372 ?        S
   Dec11   0:00 /usr/sbin/apache
root        374  0.0  1.1  7252 1496 ?        S
   Dec11   0:00 /usr/bin/gdm
root        379  0.0  1.3  7336 1672 ?        S
   Dec11   0:00 /usr/bin/gdm
root        380  0.0  6.0 30868 7676 ?        S<
   Dec11   0:04 /usr/bin/X11/X :0 -deferglyphs 16 -
root        383  0.0  0.3  1256  412 tty2     S
   Dec11   0:00 /sbin/getty -f /etc/issue.linuxlogo
root        384  0.0  0.3  1256  412 tty3     S
   Dec11   0:00 /sbin/getty -f /etc/issue.linuxlogo
root        385  0.0  0.3  1256  412 tty4     S
   Dec11   0:00 /sbin/getty -f /etc/issue.linuxlogo
root        386  0.0  0.3  1256  412 tty5     S
   Dec11   0:00 /sbin/getty -f /etc/issue.linuxlogo
root        387  0.0  0.3  1256  412 tty6     S
   Dec11   0:00 /sbin/getty -f /etc/issue.linuxlogo
gdm         394  0.0  1.9  8700 2492 ?        S
   Dec11   0:00 /usr/bin/gdmlogin --disable-sound -
root        408  0.0  1.4  4912 1832 ?        S
   Dec11   0:00 sendmail: MTA: accepting connection
root        439  0.0  0.3  1256  412 tty1     S
   Dec11   0:00 /sbin/getty -f /etc/issue.linuxlogo
www-data   6069  0.0  1.1  2944 1396 ?        S
   06:26   0:00 /usr/sbin/apache
www-data   6070  0.0  1.1  2944 1396 ?        S
   06:26   0:00 /usr/sbin/apache
www-data   6071  0.0  1.1  2944 1396 ?        S
   06:26   0:00 /usr/sbin/apache
www-data   6072  0.0  1.1  2944 1396 ?        S
   06:26   0:00 /usr/sbin/apache
www-data   6073  0.0  1.1  2944 1396 ?        S
   06:26   0:00 /usr/sbin/apache
root       7009  0.0  1.2  5704 1632 ?        S
   08:46   0:00 /usr/sbin/sshd
joao       7011  0.0  1.3  5812 1740 ?        S
   08:46   0:01 /usr/sbin/sshd
```

```
joao       7012  0.0  1.1  2620 1492 pts/0    S
   08:46   0:00 -sh
root       7591  0.0  1.2  5704 1632 ?        S
   10:14   0:00 /usr/sbin/sshd
joao       7596  0.0  1.3  5812 1740 ?        S
   10:15   0:00 /usr/sbin/sshd
joao       7597  0.0  1.0  2468 1340 pts/1    S
   10:15   0:00 -sh
joao       7660  0.0  0.8  2016 1028 pts/0    T
   10:25   0:00 top
root       7992  0.0  1.0  2472 1336 pts/1    S
   11:14   0:00 bash
root       8012  0.0  1.1  3288 1408 pts/1    R
   11:19   0:00 ps aux
```

Opções:

- a : mostra os processos criados por todos os usuários do sistema.
- x : mostra processos que não são controlados por terminal.
- u : mostra o nome de usuário que iniciou o processo e hora em que o processo foi iniciado.

O terminal onde você inicia uma tarefa é chamado de terminal que controla a tarefa.

Outra opção é utilizar `--forest` que mostra a hierarquia de processos:

Listagem 6.23: Controlando Processos

```
$ ps x --forest
  PID TTY      STAT   TIME COMMAND
 8100 ?        S      0:00 /usr/sbin/sshd
 8101 pts/0    S      0:00  \_ -sh
 8168 pts/0    R      0:00      \_ ps x --forest
```

Cada processo no Linux tem uma prioridade. Esta prioridade determina a velocidade relativa que o processo irá rodar em seu sistema. Você pode mudar a prioridade de um processo com o comando **nice**. Quanto menor o seu valor, maior a prioridade do processo (varia de -20 a 19). Os processo recém-criados herdam do pai o valor do **nice**.

Crie o programa abaixo:

Listagem 6.24: Controlando Processos

```
$ vi teste.c
// digite o código abaixo
#include <stdio.h>
```

```
main()
{
        while (1)
        {

        }
}
```

Este código cria um programa com um loop infinito. Agora compile o mesmo.

Listagem 6.25: Controlando Processos

```
$ gcc -c teste.c
$ gcc -o teste teste.o
// pronto agora temos o arquivo executável teste
// agora copie o executável teste para teste1
$ cp teste teste1
```

Humm... precisamos colocar os programas rodando em background. Como se faz isto?

Listagem 6.26: Controlando Processos

```
$ ./teste &
$ ./teste1 &
// vamos listar os processos
$ ps
  PID TTY          TIME CMD
 8224 pts/0    00:00:00 sh
 8241 pts/0    00:04:14 teste
 8250 pts/0    00:02:38 teste1
 8301 pts/0    00:00:00 ps
$ ps -l
  F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN   TTY
             TIME CMD
000 S  1000  8224  8223  0  75   0 -   617 wait4   pts
   /0    00:00:00 sh
000 R  1000  8241  8224 59  77   0 -   308 -       pts
   /0    00:04:49 teste
000 R  1000  8250  8224 49  80   0 -   308 -       pts
   /0    00:03:13 teste1
000 R  1000  8305  8224  0  76   0 -   822 -       pts
   /0    00:00:00 ps
```

O campo PRI indica a prioridade do processo. O processo teste tem a prioridade 77 e o processo teste1 tem a prioridade 80. Quem vai ficar mais tempo na CPU? Logicamente o processo com maior prioridade, mas os que tem a maior prioridade tem o menor número. Confuso? Vamos usar o top para verificar isto:

Listagem 6.27: Controlando Processos

```
11:56:23 up 21:09,  1 user,  load average: 1.99, 1.76,
    0.99
39 processes: 35 sleeping, 4 running, 0 zombie, 0
   stopped
CPU states:  100.0% user,   0.0% system,   0.0% nice,
     0.0% idle
Mem:    126820K total,   115752K used,    11068K free,
        12228K buffers
Swap:   248968K total,     6128K used,   242840K free,
        45564K cached

  PID USER      PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM
      TIME COMMAND
 8241 joao       20   0   244  244   204 R    50.8  0.1
     6:35 teste
 8250 joao       14   0   244  244   204 R    49.2  0.1
     4:59 teste1
    1 root        8   0   484  444   424 S     0.0  0.3
       0:00 init
    2 root        9   0     0    0     0 SW    0.0  0.0
       0:00 keventd
    3 root       19  19     0    0     0 SWN   0.0  0.0
       0:00 ksoftirqd_CPU0
    4 root        9   0     0    0     0 SW    0.0  0.0
       0:06 kswapd
    5 root        9   0     0    0     0 SW    0.0  0.0
       0:00 bdflush
    6 root        9   0     0    0     0 SW    0.0  0.0
       0:01 kupdated
  133 root        9   0   628  576   496 S     0.0  0.4
      0:00 dhclient-2.2.x
  208 root        9   0   600  596   492 S     0.0  0.4
      0:03 syslogd
  211 root        9   0  1052 1044   412 S     0.0  0.8
      0:01 klogd
  243 root        9   0  1244  960   804 S     0.0  0.7
      0:11 nmbd
```

```
  245 root        9    0   1100   788    636 S      0.0   0.6
        0:00 smbd
  342 root        9    0   1120  1052    944 S      0.0   0.8
        0:00 sshd
  352 root        9    0    872   832    696 S      0.0   0.6
        0:00 xinetd
  355 nobody      8    0   1148  1028    888 S      0.0   0.8
        0:00 proftpd
  358 daemon      9    0    580   556    504 S      0.0   0.4
        0:00 atd
```

Como podemos ver, o processo teste está consumindo a maior parte da CPU. Vamos matar os dois processos para entender mais sobre prioridade.

Listagem 6.28: Controlando Processos

```
$ ps
  PID TTY          TIME CMD
 8224 pts/0    00:00:00 sh
 8241 pts/0    00:07:54 teste
 8250 pts/0    00:06:18 teste1
 8330 pts/0    00:00:00 ps
$ kill -9 8241
$ kill -9 8250
[1]-  Morto                  ./teste
 [2]+  Morto                  ./teste1
$ ps
  PID TTY          TIME CMD
 8224 pts/0    00:00:00 sh
 8332 pts/0    00:00:00 ps
```

O comando **nice** permite que ao inicializar um programa possamos determinar sua prioridade de escalonamento. Conseguimos determinar as prioridades de escalonamento do **nice** de -19 a -1.

Listagem 6.29: Controlando Processos

```
$ nice -19 ./teste
$ nice -10 ./teste
$ top
12:12:41 up 21:25,  1 user,  load average: 1.13, 0.57,
    0.71
39 processes: 35 sleeping, 4 running, 0 zombie, 0
   stopped
CPU states:   0.0% user,   0.0% system,  100.0% nice,
     0.0% idle
```

```
Mem:    126820K total,   115820K used,    11000K free,
       12228K buffers
Swap:   248968K total,     6128K used,   242840K free,
       45584K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM
        TIME COMMAND
 8451 joao      20  10   244  244   204 R N  75.0  0.1
      0:32 teste1
 8450 joao      20  19   244  244   204 R N  25.0  0.1
      0:16 teste
    1 root       8   0   484  444   424 S     0.0  0.3
          0:00 init
    2 root       9   0     0    0     0 SW    0.0  0.0
          0:00 keventd
    3 root      19  19     0    0     0 SWN   0.0  0.0
          0:00 ksoftirqd_CPU0
    4 root       9   0     0    0     0 SW    0.0  0.0
          0:06 kswapd
    5 root       9   0     0    0     0 SW    0.0  0.0
          0:00 bdflush
    6 root       9   0     0    0     0 SW    0.0  0.0
          0:01 kupdated
  133 root       9   0   628  576   496 S     0.0  0.4
        0:00 dhclient-2.2.x
  208 root       9   0   600  596   492 S     0.0  0.4
        0:03 syslogd
  211 root       9   0  1052 1044   412 S     0.0  0.8
        0:01 klogd
  243 root       9   0  1244  960   804 S     0.0  0.7
        0:11 nmbd
  245 root       9   0  1100  788   636 S     0.0  0.6
        0:00 smbd
  342 root       9   0  1120 1052   944 S     0.0  0.8
        0:00 sshd
  352 root       9   0   872  832   696 S     0.0  0.6
        0:00 xinetd
  355 nobody     8   0  1148 1028   888 S     0.0  0.8
        0:00 proftpd
  358 daemon     9   0   580  556   504 S     0.0  0.4
        0:00 atd
```

Como podemos observar na coluna NI, o programa teste1 tem agora maior prioridade de escalonamento do que o programa teste. Outra forma

de verificar é o %CPU. Quem está consumindo mais.

Vamos supor que agora desejamos mudar a prioridade de escalonamento do teste. O comando **nice** somente a prioridade quando o comando é iniciado. Assim vamos utilizar o **renice**. A faixa do **renice** varia de 0 a +20.

Listagem 6.30: Controlando Processos

```
// primeiro precisamos do PID do programa teste
$ ps
 PID TTY          TIME CMD
 8224 pts/0    00:00:00 sh
 8450 pts/0    00:01:28 teste
 8451 pts/0    00:04:09 teste1
 8471 pts/0    00:00:00 ps
// hummm é o número 8450
// vamos mudar para 9 a prioridade
$ renice 9 8450
// se der algum erro de permissão mude para o usuário
   root
# renice 9 8450
8450: old priority 19, new priority 9

$ top

12:20:39 up 21:33,  1 user,  load average: 1.99, 1.70,
    1.21
44 processes: 40 sleeping, 4 running, 0 zombie, 0
   stopped
CPU states:   0.2% user,   0.0% system,   99.8% nice,
     0.0% idle
Mem:    126820K total,   116968K used,     9852K free,
       12228K buffers
Swap:   248968K total,     6128K used,   242840K free,
       45600K cached
  PID USER      PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM
        TIME COMMAND
 8450 joao      20   9   244  244   204 R N  49.9  0.1
      2:30 teste
 8451 joao      20  10   244  244   204 R N  49.7  0.1
      6:14 teste1
 8223 joao       9   0  1768 1740  1552 R     0.1  1.3
      0:00 sshd
    1 root       8   0   484  444   424 S     0.0  0.3
         0:00 init
```

```
  2 root       9   0     0    0     0 SW     0.0  0.0
      0:00 keventd
  3 root      19  19     0    0     0 SWN    0.0  0.0
      0:00 ksoftirqd_CPU0
  4 root       9   0     0    0     0 SW     0.0  0.0
      0:06 kswapd
  5 root       9   0     0    0     0 SW     0.0  0.0
      0:00 bdflush
  6 root       9   0     0    0     0 SW     0.0  0.0
      0:01 kupdated
133 root       9   0   628  576   496 S      0.0  0.4
    0:00 dhclient-2.2.x
208 root       9   0   600  596   492 S      0.0  0.4
    0:03 syslogd
211 root       9   0  1052 1044   412 S      0.0  0.8
    0:01 klogd
243 root       9   0  1244  960   804 S      0.0  0.7
    0:11 nmbd
245 root       9   0  1100  788   636 S      0.0  0.6
    0:00 smbd
342 root       9   0  1120 1052   944 S      0.0  0.8
    0:00 sshd
352 root       9   0   872  832   696 S      0.0  0.6
    0:00 xinetd
355 nobody     8   0  1148 1028   888 S      0.0  0.8
    0:00 proftpd
```

Após algum tempo, o programa teste está consumindo a maior parte da CPU, pois alteramos a prioridade de escalonamento. Agora, mate todos estes processos antes que eles acabem com os recursos de sua máquina.

Capítulo 7

Permissão e Propriedade

Sumário

7.1 Permissão e Propriedade

O Linux herdou o conceito de permissão e propriedade de arquivos utilizado no UNIX. Os sistemas UNIX supõem que a máquina pode ser compartilhada por diferentes usuários, por este motivo, é necessário atribuir posse e permissões diferentes para cada usuário do sistema. Cada arquivo possui então um dono e permissões diferenciadas para quem é o dono e quem não é. Para verificar estas características em um arquivo, você pode usar o comando **ls** conforme o exemplo na Listagem 7.1.

Listagem 7.1: Listando o Proprietário e as Permissões dos Arquivos

```
$ ls -l
-rw-rw-r-- 1 bob users 375600 Dez  5 14:30 myfile
```

O arquivo que foi listado no exemplo na Listagem 7.1 pertence ao usuário *bob*. Este, por sua vez, está no grupo de usuários chamado *users*. As permissões do arquivo estão listadas no código `-rw-rw-r--`. O primeiro traço à esquerda significa que este é um arquivo normal, contendo qualquer tipo de dados. Um diretório teria um `d` ao invés do traço `-`.

Os próximos 9 caracteres são as permissões do arquivo. Os 3 primeiros dizem quais são as permissões do usuário dono do arquivo. Os próximos 3 dizem quais são as permissões que os usuários do grupo possuem sobre o arquivo. Por fim, os 3 últimos caracteres dizem quais são as permissões para qualquer outra pessoa.

Cada grupo de 3 caracteres versa sobre as permissões de leitura (r, read), escrita (w, write) e execução (x, execute) do arquivo, nesta ordem. O arquivo ilustrado na Listagem 7.1 possui permissão de leitura e escrita para o usuário *bob* e os membros do grupo *users*, os demais usuários possuem permissão apenas de leitura.

Cada umas das permissões é binária (tem permissão ou não tem). Desta forma, cada conjunto de permissão expresso pelos 3 caracteres pode ser representa do por um número binário com 3 bits. Assim, temos o seguinte:

`---`	$(000)_2 = 0$	todas permissões negadas
`--x`	$(001)_2 = 1$	permissão apenas de execução
`-w-`	$(010)_2 = 2$	permissão apenas de escrita
`-wx`	$(011)_2 = 3$	permissão para escrita e execução
`r--`	$(100)_2 = 4$	permissão para leitura
`r-x`	$(101)_2 = 5$	permissão para leitura e execução
`rw-`	$(110)_2 = 6$	permissão para leitura e escrita
`rwx`	$(111)_2 = 7$	permissão para leitura, escrita e execução

7.2 Alterando Permissões dos Arquivos

Vamos trabalhar com o conceito de permissões do arquivo utilizando o arquivo cidades.txt criado utilizando o código da Listagem 7.2.

Listagem 7.2: Criando arquivo cidadex.txt

```
$ touch arquivo.txt
```

Vamos verificar as permissões do arquivo utilizado o comando ls no formato longo como na Listagem 7.3.

Listagem 7.3: Criando arquivo cidadex.txt

```
$ ls -l cidades.txt
-rw-rw-r-- 1 vivas vivas 0 Sep  6 14:59 cidades.txt
```

Este arquivo tem permissões de leitura e escrita para o dono, leitura e escrita para o grupo e apenas de leitura para outros. Podemos também mudar as permissões utilizando as representações simbólicas. Para os usuários:

u para o usuário

g para o grupo

o para outros

a para todos

Tipos de permissões:

r permissão de leitura

w permissão de escrita

x permissão de execução

Vamos utilizar o símbolo de + para adicionar permissões e **-** para remover permissões. Assim, basta adicionar ao identificador do usuário (u, g, o, a) o símbolo de inserir ou remover (+ ou -) e o tipo de permissão desejada (r, w ou x). Vamos iniciar removendo as permissões de leitura para o grupo. Assim, nosso comando ficará chmod g-r como na Listagem 7.4.

Listagem 7.4: Removendo Permissão de Leitura do Grupo

```
$ chmod g-r cidades.txt
vivas@musashi:~/Livros/LivroLinux2aEdicao/Permissoes$
   ls -l cidades.txt
-rw--w-r-- 1 vivas vivas 0 Sep  6 14:59 cidades.txt
```

Agora vamos remover as permissões de leitura para o outros. Assim, nosso comando ficará chmod o-r como na Listagem 7.5.

Listagem 7.5: Removendo Permissão de Leitura para Outros

```
$ chmod o-r cidades.txt
$ ls -l cidades.txt
-rw--w---- 1 vivas vivas 0 Sep  6 14:59 cidades.txt
```

Por fim vamos remover as permissões de leitura do arquivo para o usuário. Assim, nosso comando ficará chmod u-r como na Listagem 7.6.

Listagem 7.6: Removendo Permissão de Leitura para o Usuário

```
$ chmod u-r cidades.txt
$ ls -l cidades.txt
--w--w---- 1 vivas vivas 0 Sep  6 14:59 cidades.txt
```

Agora vamos adicionar permissões novamente em nosso arquivo. Queremos adicionar permissões de leitura para usuário, grupo e outros, simultaneamente. Temos duas opções de uso do comando: chmod ugo+r ou chmod a+r . O comando a (all - todos) adiciona permissões para usuário, grupo e outros simultaneamente como na Listagem 7.7.

Listagem 7.7: Adicionando Permissões para todos

```
$ chmod a+r cidades.txt
$ ls -l cidades.txt
-rw-rw-r-- 1 vivas vivas 0 Sep  6 14:59 cidades.txt
```

O meio mais fácil para adicionar ou remover permissões para um arquivo é desta maneira. O uso de permissões binárias justifica-se quando precisamos fazer alterações em um conjunto de permissões do arquivo.

Comando	Permissões
chmod 111	–x–x–x
chmod 222	-w–w–w-
chmod 333	-wx-wx-wx
chmod 444	r–r–r–
chmod 555	r-xr-xr-x
chmod 666	rw-rw-rw-
chmod 777	rwxrwxrwx

Tabela 7.1: Exemplos de Permissões de Arquivos

Listagem 7.8: Verificando permissões do arquivo

```
$ touch arquivo.txt
```

7.3 Alterando Dono do Arquivo

Podemos trocar o dono de um arquivo, assim como as permissões. Para tanto, utilizaremos os comandos *chown* e *chmod*, respectivamente.

Listagem 7.9: Trocando o Dono de um Arquivo

```
$ chown john:users2 myfile
$ ls -l myfile
-rw-rw-r-- 1 john users2 375600 Dez  5 14:30 myfile
```

Listagem 7.10: Trocando as Permissões de um Arquivo

```
$ chmod 777 myfile
$ ls -l myfile
-rwxrwxrwx 1 john users2 375600 Dez  5 14:30 myfile
$ chmod 600 myfile
$ ls -l myfile
-rw------- 1 john users2 375600 Dez  5 14:30 myfile
```

Os seguintes exemplos na Listagem 7.11 ilustram algumas possíveis utilizações destas representações para alterar as permissões de um arquivo ou diretório.

Listagem 7.11: Exemplos de Utilização do chmod

```
# adicionar a permissao de execucao apenas ao usuario
$ chmod u+x arquivo

# adicionar multiplas permissoes (por exemplo, leitura
    e execucao)
$ chmod u+rx arquivo

# adicionar permissoes diferentes a usuario (permissao
    de leitura) e (permissao de grupo) grupo
$ chmod u+r,g+x arquivo

# remover permissoes (leitura e execucao)
$ chmod u-rx

# adicionar a permissao de execucao a todos os
   usuarios
$ chmod a+x
```

Capítulo 8

Gerenciando Usuários

Sumário

8.1 Listando Todos os Usuários do Sistema

Para listar todos os usuários do sistema Linux basta verificar quais estão listados no arquivo */etc/passwd* utilizando, para tanto, o comando **cat** como na Listagem 8.1, na qual utiliza-se o **head** para limitar aos 10 primeiros.

Listagem 8.1: Listando Todos os Usuários do Linux

```
$ cat /etc/passwd | head -10
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

Ou ainda, podemos utilizar o **cut**, como na Listagem 8.2, onde utiliza-se novamente o **head** para limitar aos 10 primeiros.

Listagem 8.2: Listando Todos os Usuários do Linux

```
$ cut -d: -f1 /etc/passwd | head -10
root
daemon
bin
sys
sync
games
man
lp
mail
news
```

8.2 Listando Grupos

Para listar os grupos disponíveis no Linux, verificar o arquivo */etc/group*, por exemplo, utiliza-se o comando **cat** , como na Listagem 8.3.

Listagem 8.3: Listando Todos os Usuários do Linux

```
$ cat /etc/group | head -10
root:x:0:
```

```
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,leoca
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
```

8.3 Adicionando Usuários

Para adicionar um usuário, basta utilizar o comando **useradd** ou **adduser** . A sintaxe para adicionar usuário está apresentada na Listagem 8.4. Esta operação é privilegiada e só deve ser utilizada como **root**.

Listagem 8.4: Adicionando Usuários

```
sudo adduser aluno
[sudo] password for vivas:
Adding user 'aluno' ...
Adding new group 'aluno' (1017) ...
Adding new user 'aluno' (1012) with group 'aluno' ...
Creating home directory '/home/aluno' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for aluno
Enter the new value, or press ENTER for the default
        Full Name []: Aluno
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

8.4 Definindo Senha para Novos Usuários

Para adicionar ou mudar a senha de um usuário, basta utilizar o comando **passwd** . A sintaxe para adicionar usuário é apresentada na Listagem 8.5.

Listagem 8.5: Definindo Senha de Usuário

```
$ passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

8.5 Apagando uma Conta de Usuário

Para apagar uma conta de usuário, basta utilizar o comando **userdel** . A sintaxe para adicionar usuário é apresentada na Listagem 8.6.

Listagem 8.6: Apagando Usuários

```
$ sudo userdel aluno
```

Caso deseje apagar também o seu diretório, utilize o comando na Listagem 8.7.

Listagem 8.7: Apagando Usuários e Arquivos do Usuário

```
$ userdel -r aluno
```

8.6 Modificando Conta de Usuário

Para modificar uma conta de usuário, basta utilizar o comando **usermod** . A sintaxe para inserir uma data para a conta do usuário expirar é apresentada na Listagem 8.8.

Listagem 8.8: Modificando Conta do Usuário

```
$ sudo usermod -e 2015-11-25 aluno
```

8.7 Adicionando um Novo Grupo

Para adicionar um grupo basta utilizar o comando **groupadd** . A sintaxe para inserir um novo grupo é apresentada na Listagem 8.9.

Listagem 8.9: Adicionando um Novo Grupo

```
$ sudo groupadd professores
```

8.8 Deletando um Grupo

Para apagar um grupo basta utilizar o comando **groupdel** . A sintaxe para apagar um grupo é apresentada na Listagem 8.10.

Listagem 8.10: Apagando um Grupo

```
$ sudo groupdel professores
```

8.9 Modificando um Grupo

Para modificar um grupo, basta utilizar o comando **groupmod** . A sintaxe para modificar um grupo é apresentada na Listagem 8.11. Com este comando, renomeamos o grupo professores para funcionários.

Listagem 8.11: Modificando um Grupo

```
$ sudo groupmod -n  professores funcionarios
```

Capítulo 9

Comandos para Redes de Computadores

Sumário

9.1 Instalação

Para executar os comandos deste capítulo é necessário a instalação do pacote net-tools. Algumas distribuições já possuem este pacote instalado. Para realizar a instalação, utilize o procedimento na Listagem 9.1.

Listagem 9.1: Instalação do net-tools

```
apt install net-tools
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os NOVOS pacotes a seguir serão instalados:
  net-tools
0 pacotes atualizados, 1 pacotes novos instalados, 0 a
    serem removidos e 1 não atualizados.
É preciso baixar 248 kB de arquivos.
Depois desta operação, 1.002 kB adicionais de espaço
   em disco serão usados.
Obter:1 http://ftp.br.debian.org/debian buster/main
   amd64 net-tools amd64 1.60+git20180626.aebd88e-1
   [248 kB]
Baixados 248 kB em 0s (1.224 kB/s)
A seleccionar pacote anteriormente não seleccionado
   net-tools.
(Lendo banco de dados ... 316554 ficheiros e directó
   rios actualmente instalados.)
A preparar para desempacotar .../net-tools_1.60+
   git20180626.aebd88e-1_amd64.deb ...
A descompactar net-tools (1.60+git20180626.aebd88e-1)
   ...
Configurando net-tools (1.60+git20180626.aebd88e-1)
   ...
A processar 'triggers' para man-db (2.8.5-2)
```

9.2 Comando hostname

Quer descobrir o nome de sua máquina? Uma maneira simples é entrar no terminal e você conseguirá visualizar. Como exemplo, meu login recebe **vivaszafu:$**, neste caso estou logado com o usuário **vivas** na máquina **zafu**.

Você pode utilizar também o comando **hostname** apresentado na Listagem 9.2

Listagem 9.2: Comando hostname

```
$ hostname
masamune
```

O nome da máquina é armazenado no arquivo **hosts** que fica no diretório **etc**. Assim, podemos obter o nome da máquina utilizando o comando como na Listagem 9.3.

Listagem 9.3: Nome da Máquina com cat

```
$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       masamune

# The following lines are desirable for IPv6 capable
   hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

9.3 Comando e Tabela ARP

As comunicações nas redes locais necessitam do endereço MAC, mas geralmente temos o endereço IP da outra máquina. Assim, cada computador possui um servidor ARP (Address Resolution Protocol) que fornece o endereço MAC de nossa máquina quando solicitado.

O protocolo ARP é responsável por receber um pacote com o endereço IP e enviar para o destinatário o endereço MAC. Quer saber a tabela ARP do seu computador? Isto é, os computadores que de alguma forma você entrou em contato? Este comando funciona no MAC, no Linux e até no Windows, Listagem 9.4. Através do comando **arp**, é possível visualizar a tabela ARP. Algumas distribuições exigem que este comando seja utilizado pelo super-usuário.

Listagem 9.4: Tabela ARP

```
$ arp -a
? (192.168.0.1) at 1c:7e:e5:46:92:e7 on en1 ifscope [
   ethernet]
? (192.168.0.100) at 4c:e6:76:be:ee:a9 on en1 ifscope
   [ethernet]
```

9.4 Verificando o Endereço IP de sua Máquina

Para verificar seu endereço IP basta utilizar o comando **ifconfig** . Como pode ser visto, aparecem duas interfaces neste comando:

- eno1 : neste caso, temos uma interface para rede cabeada;
- lo: loopback interface utilizada para realização de testes. Ao enviar um pacote para esta interface o pacote não vai para rede externa.

A Listagem 9.5 ilustra o resultado do comando **ifconfig**.

9.4.1 Verificando Endereço IP

Listagem 9.5: Verificando o Endereço IP

```
# ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   1500
        inet 192.168.1.104  netmask 255.255.255.0
           broadcast 192.168.1.255
        inet6 fe80::f24d:a2ff:fee2:fb5a  prefixlen 64
            scopeid 0x20<link>
        ether f0:4d:a2:e2:fb:5a  txqueuelen 1000  (
           Ethernet)
        RX packets 140898  bytes 140297327 (133.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 98391  bytes 16417138 (15.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
        device interrupt 21  memory 0xf7fe0000-
           f8000000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback Local)
        RX packets 521  bytes 39576 (38.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 521  bytes 39576 (38.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
```

9.5 Habilitando e Desabilitando a Interface de Rede

Para habilitar ou desabilitar uma interface de rede utilizamos o comando **ifconfig** . Primeiro vamos verificar o status da interface de rede Ethernet, Listagem 9.6.

Listagem 9.6: Verificando Status da Rede Ethernet

```
~$ ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   1500
        inet 200.131.20.201  netmask 255.255.255.240
           broadcast 200.131.20.207
        inet6 fe80::862b:2bff:fe79:b64f  prefixlen 64
            scopeid 0x20<link>
        ether 84:2b:2b:79:b6:4f  txqueuelen 1000  (
           Ethernet)
        RX packets 6178516  bytes 4477239473 (4.4 GB)
        RX errors 0  dropped 12200  overruns 0  frame
           0
        TX packets 3034496  bytes 918584412 (918.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
```

Para desabilitar, utilizamos o comando da Listagem 9.6. Para verificar vamos utilizar o comando **ifconfig** na Listagem 9.7. Existem duas abordagens para utilizar este comando: a) algumas distribuições necessitam que você esteja logado como root (Debian); b) outras distribuições você precisará digitar sudo antes do comando (Ubuntu).

Listagem 9.7: Desabilitando a Interface de Rede Ethernet

```
root@masamune:~# ifconfig eno1 down
root@masamune:~# ifconfig eno1
eno1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.1.104  netmask 255.255.255.0
           broadcast 192.168.1.255
        ether f0:4d:a2:e2:fb:5a  txqueuelen 1000  (
           Ethernet)
        RX packets 141187  bytes 140470648 (133.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 98591  bytes 16444036 (15.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
        device interrupt 21  memory 0xf7fe0000-
           f8000000
```

Para verificar se a interface está desabilitada, proceda com o comando **ifconfig** como na Listagem 9.8. Na Listagem anterior, percebem-se que os flags UP não estão habilitados quando digitamos o comando.

Listagem 9.8: Verificando a Ação Realizada na Listagem 9.7

```
# ifconfig
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback Local)
        RX packets 705  bytes 54408 (53.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 705  bytes 54408 (53.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
```

Para habilitar, utilizamos o comando da Listagem 9.9 e, para verificar, vamos utilizar o comando **ifconfig**, na Listagem 9.10.

Listagem 9.9: Habilitando a Interface de Rede Ethernet

```
root@masamune:~# ifconfig eno1 up
```

Listagem 9.10: Verificando a Ação Realizada na Listagem 9.9

```
root@masamune:~# ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   1500
        inet 192.168.1.104  netmask 255.255.255.0
           broadcast 192.168.1.255
        inet6 fe80::f24d:a2ff:fee2:fb5a  prefixlen 64
            scopeid 0x20<link>
        ether f0:4d:a2:e2:fb:5a  txqueuelen 1000  (
           Ethernet)
        RX packets 174189  bytes 179466863 (171.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 117638  bytes 18574493 (17.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
        device interrupt 21  memory 0xf7fe0000-
           f8000000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback Local)
        RX packets 788  bytes 61140 (59.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 788  bytes 61140 (59.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
```

9.6 Alterando a MTU de uma Interface

MTU (*Maximum Transmission Unit*) é o tamanho do maior datagrama que pode ser transmitido em uma determinada rede. As redes Ethernet modernas utilizam o tamanho de 1500 bytes. Para listar a MTU utilizada usamos o comando **ifconfig** da Listagem 9.11. Para verificar isto, utilizamos o comando **ifconfig** .

Listagem 9.11: Verificando MTU da Rede Ethernet

```
root@masamune:~# ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   1500
        inet 192.168.1.104  netmask 255.255.255.0
           broadcast 192.168.1.255
        inet6 fe80::f24d:a2ff:fee2:fb5a  prefixlen 64
            scopeid 0x20<link>
        ether f0:4d:a2:e2:fb:5a  txqueuelen 1000  (
           Ethernet)
        RX packets 174267  bytes 179479126 (171.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 117702  bytes 18585549 (17.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
        device interrupt 21  memory 0xf7fe0000-
           f8000000
```

É possível alterar a MTU utilizando o comando **ifconfig** da Listagem 9.12.

Listagem 9.12: Alterando o MTU da Placa de Rede

```
root@masamune:~# ifconfig eno1 mtu 600
root@masamune:~# ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   600
        inet 192.168.1.104  netmask 255.255.255.0
           broadcast 192.168.1.255
        ether f0:4d:a2:e2:fb:5a  txqueuelen 1000  (
           Ethernet)
        RX packets 174978  bytes 180012523 (171.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 118149  bytes 18626402 (17.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
```

```
        device interrupt 21  memory 0xf7fe0000-
           f8000000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback Local)
        RX packets 790  bytes 61284 (59.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 790  bytes 61284 (59.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
```

9.7 Alterando Endereço IP

Se quiser alterar seu endereço IP, basta utilizar o comando **ifconfig**. O comando recebe como parâmetros: interface, endereço IP e máscara. A Listagem 9.13 apresenta o endereço IP atual da interface eth0.

Listagem 9.13: Verificando o Endereço IP

```
root@masamune:~# ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   1500
        inet 192.168.1.104  netmask 255.255.255.0
           broadcast 192.168.1.255
        inet6 fe80::f24d:a2ff:fee2:fb5a  prefixlen 64
            scopeid 0x20<link>
        ether f0:4d:a2:e2:fb:5a  txqueuelen 1000  (
           Ethernet)
        RX packets 179014  bytes 183338695 (174.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 120576  bytes 18915193 (18.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
        device interrupt 21  memory 0xf7fe0000-
           f8000000
```

A Listagem 9.14 apresenta o comando de alteração do endereço IP.

Listagem 9.14: Alterando Endereço IP

```
root@masamune:~# ifconfig eno1 192.158.1.110 netmask
   255.255.255.0
```

```
root@masamune:~# ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   1500
        inet 192.158.1.110  netmask 255.255.255.0
           broadcast 192.158.1.255
        inet6 fe80::f24d:a2ff:fee2:fb5a  prefixlen 64
            scopeid 0x20<link>
        ether f0:4d:a2:e2:fb:5a  txqueuelen 1000  (
           Ethernet)
        RX packets 179225  bytes 183375049 (174.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 120768  bytes 18940932 (18.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
           collisions 0
        device interrupt 21  memory 0xf7fe0000-
           f8000000
```

9.8 Comando ping

O comando **ping** serve para fazer verificações sobre o status de funcionamento de computadores em uma rede. Com ele podemos medir o tempo de ida e volta (*round time trip*) que um pacote demora para ir do seu host para outro. Você pode usar tanto o endereço IP do host ou o endereço Web. A Listagem 9.15 mostra o funcionamento do comando **ping**. Podemos passar um endereço IP, como ilusrtado na Listagem 9.15, ou utilizar como argumento um endereço Web, como ilustrado na Listagem 9.16. Para interromper o comando, basta digitar <Ctrl+C>. Quando interromper o comando, serão mostradas as estatísticas dos testes realizados.

Listagem 9.15: Comando ping

```
root@masamune:~# ping 10.1.212.1
PING 10.1.212.1 (10.1.212.1) 56(84) bytes of data.
64 bytes from 10.1.212.1: icmp_seq=1 ttl=254 time=2.08 ms
64 bytes from 10.1.212.1: icmp_seq=2 ttl=254 time=1.41 ms
64 bytes from 10.1.212.1: icmp_seq=3 ttl=254 time=1.48 ms
64 bytes from 10.1.212.1: icmp_seq=4 ttl=254 time=1.56 ms
64 bytes from 10.1.212.1: icmp_seq=5 ttl=254 time=2.49 ms
64 bytes from 10.1.212.1: icmp_seq=6 ttl=254 time=1.41 ms
64 bytes from 10.1.212.1: icmp_seq=7 ttl=254 time=1.46 ms
64 bytes from 10.1.212.1: icmp_seq=8 ttl=254 time=1.39 ms
64 bytes from 10.1.212.1: icmp_seq=9 ttl=254 time=1.89 ms
^C
--- 10.1.212.1 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 20ms
rtt min/avg/max/mdev = 1.389/1.684/2.492/0.368 ms
```

Listagem 9.16: Exemplo do Comando ping

```
# ping www.ufu.br
PING www.ufu.br (200.19.146.58) 56(84) bytes of data.
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=1 ttl=53 time=18.4 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=2 ttl=53 time=18.4 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=3 ttl=53 time=17.5 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=4 ttl=53 time=17.6 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=5 ttl=53 time=20.7 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=6 ttl=53 time=17.6 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=7 ttl=53 time=19.5 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=8 ttl=53 time=22.3 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=9 ttl=53 time=19.6 ms
^C
--- www.ufu.br ping statistics ---
9 packets transmitted, 9 received, 0% packet loss,
   time 21ms
rtt min/avg/max/mdev = 17.476/19.054/22.303/1.543 ms
```

Você pode utilizar a opção **-c** com o **ping**, que especifica o número de pacotes enviados pelo ping. A Listagem 9.17 apresenta o comando para 10 tentativas.

Listagem 9.17: Comando ping

```
ping -c 10 www.ufu.br
PING www.ufu.br (200.19.146.58) 56(84) bytes of data.
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=1 ttl=53 time=20.7 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=2 ttl=53 time=17.2 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=3 ttl=53 time=17.6 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=5 ttl=53 time=51.7 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=6 ttl=53 time=18.5 ms
```

```
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=7 ttl=53 time=17.6 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=8 ttl=53 time=17.4 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=9 ttl=53 time=17.4 ms
64 bytes from bulma.dr.ufu.br (200.19.146.58):
   icmp_seq=10 ttl=53 time=17.4 ms

--- www.ufu.br ping statistics ---
10 packets transmitted, 9 received, 10% packet loss,
   time 28ms
rtt min/avg/max/mdev = 17.211/21.725/51.652/10.631 ms
```

Quer controlar o intervalo de tempo entre os pacotes enviados? Utilize a opção **-i** como mostrado na Listagem 9.18. Vamos aproveitar e utilizar a opção de enviar 3 pacotes.

Listagem 9.18: Comando ping com Opção de Tempo

```
$ ping -i 5 -c 3 www.google.com
PING www.google.com (172.217.30.68) 56(84) bytes of
   data.
64 bytes from gru06s34-in-f4.1e100.net (172.217.30.68)
   : icmp_seq=1 ttl=47 time=22.9 ms
64 bytes from gru06s34-in-f4.1e100.net (172.217.30.68)
   : icmp_seq=2 ttl=47 time=20.10 ms
64 bytes from gru06s34-in-f4.1e100.net (172.217.30.68)
   : icmp_seq=3 ttl=47 time=21.5 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss,
   time 20ms
rtt min/avg/max/mdev = 20.992/21.806/22.902/0.813 ms
```

9.9 Descobrir endereço IP de um Determinado Host

Muitas vezes, precisamos descobrir o endereço IP de um determinado host. Para realizar esta tarefa utilizamos o comando **host** . A Listagem 9.19 mostra como descobrir o endereço IP do Google.

Listagem 9.19: Descobrindo o Endereço Ip de um Host

```
# host www.google.com
www.google.com has address 172.217.30.68
```

```
www.google.com has IPv6 address 2800:3f0
   :4004:801::2004
```

9.10 Informações sobre Domínios

9.10.1 Comando dig

Uma maneira de obter informações sobre domínios é utilizar o comando **dig** . Verifique se o comando está disponível em sua distribuição ou realize o procedimento de instalação, como na Listagem 9.20.

Listagem 9.20: Instalação do dnsutils

```
# apt install dnsutils
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
The following additional packages will be installed:
  libirs161
Pacotes sugeridos:
  rblcheck
Os NOVOS pacotes a seguir serão instalados:
  dnsutils libirs161
0 pacotes atualizados, 2 pacotes novos instalados, 0 a
    serem removidos e 1 não atualizados.
É preciso baixar 602 kB de arquivos.
Depois desta operação, 1.023 kB adicionais de espaço
   em disco serão usados.
Você quer continuar? [S/n] S
Obter:1 http://ftp.br.debian.org/debian buster/main
   amd64 libirs161 amd64 1:9.11.5.P4+dfsg-5.1 [237 kB]
Obter:2 http://ftp.br.debian.org/debian buster/main
   amd64 dnsutils amd64 1:9.11.5.P4+dfsg-5.1 [365 kB]
Baixados 602 kB em 6s (109 kB/s)
A seleccionar pacote anteriormente não seleccionado
   libirs161:amd64.
(Lendo banco de dados ... 316805 ficheiros e directó
   rios actualmente instalados.)
A preparar para desempacotar .../libirs161_1%3a9.11.5.
   P4+dfsg-5.1_amd64.deb ...
A descompactar libirs161:amd64 (1:9.11.5.P4+dfsg-5.1)
   ...
A seleccionar pacote anteriormente não seleccionado
   dnsutils.
```

```
A preparar para desempacotar .../dnsutils_1%3a9.11.5.
   P4+dfsg-5.1_amd64.deb ...
A descompactar dnsutils (1:9.11.5.P4+dfsg-5.1) ...
Configurando libirs161:amd64 (1:9.11.5.P4+dfsg-5.1)
   ...
Configurando dnsutils (1:9.11.5.P4+dfsg-5.1) ...
A processar 'triggers' para libc-bin (2.28-10) ...
A processar 'triggers' para man-db (2.8.5-2) ...
```

A sintaxe é bem simples, pois basta usar como parâmetro o domínio desejado. A Listagem 9.21 ilustra a utilização do comando.

Listagem 9.21: Descobrir Informações sobre um Domínio

```
dig www.google.com

; <<>> DiG 9.11.5-P4-5.1-Debian <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
   39638
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
    ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         197     IN      A
   172.217.28.228

;; Query time: 2 msec
;; SERVER: 10.253.16.5#53(10.253.16.5)
;; WHEN: ter set 17 16:50:38 -03 2019
;; MSG SIZE  rcvd: 59
```

9.10.2 Comando nslookup

Outra maneira é utilizar o comando **nslookup** . A Listagem 9.22 ilustra a utilização do comando.

Listagem 9.22: Utilizando o comando nslookup

```
 nslookup www.google.com
```

```
Server:         10.253.16.5
Address:        10.253.16.5#53

Non-authoritative answer:
Name:   www.google.com
Address: 172.217.28.228
Name:   www.google.com
Address: 2800:3f0:4004:807::2004
```

9.11 Traçando caminhos de um host a outro

O comando **traceroute** é uma ferramenta para imprimir os caminhos de seu host até um destino. Ele mostra todos os roteadores que o pacote enviado passa e imprime informações sobre o tempo decorrido. A Listagem 9.23 mostra como instalar o comando.

Listagem 9.23: Instalação do Traceroute

```
$ sudo apt install traceroute
Password:
```

Para usar, basta inserir o destino como na Listagem 9.24. Alguns roteadores são programados para não responder, e por isto apresentam o símbolo * . Outra situação é ocorrência de perda de pacotes.

Listagem 9.24: Rotas

```
$ traceroute www.google.com
traceroute to www.google.com (172.217.29.132), 30 hops max, 60
    byte packets
 1  _gateway (200.131.20.193)  0.989 ms  0.983 ms  1.058 ms
 2  core.rede.ufvjm.edu.br (200.131.20.241)  0.479 ms  0.676 ms
     0.859 ms
 3  200.131.20.117 (200.131.20.117)  0.221 ms  0.227 ms  0.195
     ms
 4  200.131.20.30 (200.131.20.30)  0.333 ms  0.323 ms  0.293 ms
 5  heimdall.rede.ufvjm.edu.br (200.131.20.1)  0.883 ms  0.871
     ms  0.854 ms
 6  200.131.21.190 (200.131.21.190)  1.343 ms  1.238 ms  1.213
     ms
 7  ufvjm-2-gw.pop-mg.rnp.br (200.19.156.253)  6.475 ms  6.417
     ms  7.065 ms
 8  almeirao.pop-mg.rnp.br (200.131.0.3)  6.646 ms couve.pop-mg.
     rnp.br (200.131.0.2)  7.064 ms  6.865 ms
 9  200.143.255.173 (200.143.255.173)  6.221 ms mg-lanmg.bkb.rnp
     .br (200.143.253.161)  6.226 ms 200.143.255.173
     (200.143.255.173)  6.149 ms
10  170.79.213.0 (170.79.213.0)  14.897 ms  14.878 ms  14.904 ms
11  170.79.213.38 (170.79.213.38)  14.961 ms  14.948 ms as15169.
    saopaulo.sp.ix.br (187.16.218.58)  17.279 ms
```

```
12  as15169.riodejaneiro.rj.ix.br (45.6.52.32)  15.205 ms
    14.313 ms 170.79.213.38 (170.79.213.38)  14.240 ms
13  108.170.251.67 (108.170.251.67)  18.943 ms 172.253.71.29
    (172.253.71.29)  16.792 ms as15169.riodejaneiro.rj.ix.br
    (45.6.52.32)  14.436 ms
14  209.85.249.228 (209.85.249.228)  20.495 ms 209.85.254.250
    (209.85.254.250)  19.496 ms 108.170.251.83 (108.170.251.83)
    19.509 ms
15  216.239.54.142 (216.239.54.142)  19.925 ms 209.85.254.98
    (209.85.254.98)  19.779 ms 172.253.67.189 (172.253.67.189)
    19.779 ms
16  gru06s47-in-f4.1e100.net (172.217.29.132)  16.509 ms
    108.170.245.161 (108.170.245.161)  19.864 ms 172.253.67.192
    (172.253.67.192)  19.894 ms
```

9.11.1 Descobrindo o Endereço do seu Roteador sem Fio

Uma das maneiras é utilizar o comando traceroute. O primeiro salto sempre será o IP do seu roteador sem fio. Verifique a Listagem 9.25. Neste caso, enviamos o comando para dois destinos diferentes e observamos que os dois sempre no primeiro salto passam pelo o IP 192.168.0.1, que é o do roteador sem fio.

Listagem 9.25: Endereço do Roteador sem Fio

```
vivas@zafu:~$ traceroute www.ufmg.br
traceroute to www.ufmg.br (150.164.250.1), 30 hops max
   , 60 byte packets
 1  192.168.0.1 (192.168.0.1)  1.030 ms  1.179 ms
    2.036 ms
^C
vivas@zafu:~$ traceroute www.mit.edu
traceroute to www.mit.edu (23.65.134.151), 30 hops max
   , 60 byte packets
 1  192.168.0.1 (192.168.0.1)  1.106 ms  1.415 ms
    1.372 ms
```

9.12 Comando tracepath

O comando **tracepath** é similar ao comando **traceroute**, mas possui opções menos complicadas. Para utilizar o comando siga a sintaxe da Listagem 9.26.

Listagem 9.26: Rotas com tracepath

```
$ tracepath www.ufmg.br
 1?:  [LOCALHOST]                      pmtu 1500
 1:   _gateway                                           1.167ms
 1:   _gateway                                           0.992ms
 2:   core.rede.ufvjm.edu.br                             0.642ms
 3:   200.131.20.117                                     0.500ms
```

```
 4:   200.131.20.30                               0.602ms
 5:   heimdall.rede.ufvjm.edu.br                  0.708ms
 6:   200.131.21.190                              0.994ms
 7:   ufvjm-2-gw.pop-mg.rnp.br                    7.989ms
 8:   tropeiro.pop-mg.rnp.br                      6.662ms
 9:   pop-mg-10g.rede.ufmg.br                    12.649ms
10:   pop.central-core.rede.ufmg.br               7.482ms
11:   cecom-gw.rede.ufmg.br                       7.815ms
12:   www.ufmg.br                                 7.156ms !H
```

9.13 Comando netstat

O comando **netstat** é uma ferramenta essencial para administradores de rede. Ele possibilita fazer rastreamento das portas que são utilizadas no seu computador. A Listagem 9.27 apresenta estatística dos protocolos.

Listagem 9.27: Estatísticas de Rede com netstat

```
vivas@zafu:~$ netstat -s
Ip:
    14476 total de pacotes recebidos
    0 encaminhado
    0 pacotes de entrada descartados
    14470 pacotes de entrada entregues
    10142 requisicoes enviadas
Icmp:
    318 mensagens ICMP recebidas
    0 mensagens ICMP de entrada com problemas.
    Histograma de entrada ICMP:
        destino inalcancavel: 96
        tempo expirou em transito: 222
    434 mensagens ICMP enviadas
    0 mensagens ICMP falharam
    Histograma de saida ICMP
        destino inalcancavel: 434
IcmpMsg:
        InType3: 96
        InType11: 222
        OutType3: 434
Tcp:
    34 conexoes ativas abertas
    2 conexoes passivas abertas
    0 tentantivas de conexao que falharam
    10 reinicios de conexoes recebidos
    3 conexoes estabelecidas
    9272 segmentos recebidos
    7445 segmentos enviados
    11 segmentos retransmitidos
```

```
    0 segmentos invalidos recebidos
    30 reinicios enviados
Udp:
    2266 pacotes recebidos
    354 pacotes recebidos para uma porta desconhecida
    0 erros na recepção de pacotes
    2269 pacotes enviados
UdpLite:
TcpExt:
    3 soquetes TCP concluiram o tempo de espera mais r
       ápido que o normal
    95 acks retardados enviados
    1 confirmacoes adiadas foram novamente adiadas
       devido a um soquete bloqueado
    6 pacotes diretamente enfileirados em recebmsg pre
       -fila.
    6748 bytes directly received in process context
       from prequeue
    3656 cabecalhos de pacotes previstos
    4 cabecalhos de pacote previstos e diretamente
       enfileirados ao usuário
    2156 acknowledgments not containing data payload
       received
    1341 reconhecimentos preditos
    2 outras expiracoes de tempo TCP
    6 conexoes resetadas devido a dados não esperados
    10 conexoes restauradas por cancelamento do usuá
       rio
    2 conexoes abortadas por tempo expirado
    IPReversePathFilter: 5
    TCPRcvCoalesce: 572
    TCPOFOQueue: 603
IpExt:
    InNoRoutes: 1
    InMcastPkts: 754
    OutMcastPkts: 185
    InBcastPkts: 2341
    OutBcastPkts: 7
    InOctets: 6692920
    OutOctets: 1130339
    InMcastOctets: 187110
    OutMcastOctets: 19647
    InBcastOctets: 1011354
    OutBcastOctets: 328
```

A Listagem 9.28 apresenta a maneira de obter estatísticas das interfaces de rede.

Listagem 9.28: Comando netstat -i

```
$ netstat -i
Tabela de Interfaces do Kernel
Iface       MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK
   TX-ERR TX-DRP TX-OVR Flg
eno1       1500   224131      0      0 0         148375
        0      0      0 BMRU
lo        65536      842      0      0 0            842
        0      0      0 LRU
```

9.13.1 Tabela de Roteamento

Para visulizar a tabela de roteamento de um host podemos utilizar também o comando **netstat**. A Listagem 9.29 apresenta o comando.

Listagem 9.29: Visualizando Tabela de Roteamento com netstat

```
vivas@zafu:~$ netstat -rn
Tabela de Roteamento IP do Kernel
Destino         Roteador        MascaraGen.     Opcoes
     MSS Janela  irtt Iface
0.0.0.0         192.168.0.1     0.0.0.0          UG
          0 0          0 eth0
169.254.0.0     0.0.0.0         255.255.0.0      U
           0 0          0 eth0
192.168.0.0     0.0.0.0         255.255.255.0    U
           0 0          0 eth0
```

9.14 Network Mapper

O comando **nmap** é uma ferramenta excelente para fazer varreduras em redes de computadores.

9.14.1 Instalação

O processo de instalação é bem simples e pode ser visualizado na Listagem 9.30.

Listagem 9.30: Instalação do nmap

```
$ sudo apt-get install nmap
Password:
```

9.14.2 Analisando portas abertas

O comando da Listagem 9.31 apresenta a versão básica do comando para listar as portas abertas de um determinado domínio.

Listagem 9.31: Verificando Portas Abertas

```
$ nmap www.ufmg.br

Starting Nmap 7.60 ( https://nmap.org ) at 2019-09-18
   15:56 -03
Nmap scan report for www.ufmg.br (150.164.250.1)
Host is up (0.017s latency).
Not shown: 989 filtered ports
PORT      STATE  SERVICE
21/tcp    open   ftp
80/tcp    open   http
443/tcp   open   https
40193/tcp closed unknown
40911/tcp closed unknown
41511/tcp closed unknown
42510/tcp closed caerpc
44176/tcp closed unknown
44442/tcp closed coldfusion-auth
44443/tcp closed coldfusion-auth
44501/tcp closed unknown

Nmap done: 1 IP address (1 host up) scanned in 11.25
   seconds
```

9.14.3 Comando nmap com opção de mais informações

Se quiser mais informações sobre o procedimento, utilize a opção **-v** como na Listagem 9.32.

Listagem 9.32: Comando nmap com opção -v

```
$ nmap -v localhost

Starting Nmap 7.60 ( https://nmap.org ) at 2019-09-18
   15:57 -03
Initiating Ping Scan at 15:57
Scanning localhost (127.0.0.1) [2 ports]
Completed Ping Scan at 15:57, 0.00s elapsed (1 total
   hosts)
Initiating Connect Scan at 15:57
```

```
Scanning localhost (127.0.0.1) [1000 ports]
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 80/tcp on 127.0.0.1
Discovered open port 111/tcp on 127.0.0.1
Discovered open port 8000/tcp on 127.0.0.1
Completed Connect Scan at 15:57, 0.04s elapsed (1000
   total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00026s latency).
Other addresses for localhost (not scanned): ::1
rDNS record for 127.0.0.1: localhost.localdomain
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
111/tcp  open  rpcbind
8000/tcp open  http-alt

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.10
   seconds
```

9.14.4 Rastreando Múltiplos Hosts

Para rastrear múltiplos hosts, basta passar os endereços de IPs desejados. A Listagem 9.33 ilustra o procedimento para dois hosts.

Listagem 9.33: Rastreando Múltiplos Hosts.numbers

```
vivas@zafu:~$ nmap 192.168.0.1 192.168.0.104

Starting Nmap 5.21 ( http://nmap.org ) at 2013-12-17
   15:42 BRST
Nmap scan report for 192.168.0.1
Host is up (0.0074s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
53/tcp open  domain
80/tcp open  http

Nmap scan report for 192.168.0.104
Host is up (0.00018s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
```

```
22/tcp open  ssh

Nmap done: 2 IP addresses (2 hosts up) scanned in 0.47
    seconds
```

É possível rastrear múltiplos domínios como na Listagem 9.34.

Listagem 9.34: Rastreando Múltiplos Domínios

```
vivas@zafu:~$ nmap www.google.com www.facebook.com

Starting Nmap 5.21 ( http://nmap.org ) at 2013-12-17
   15:45 BRST
Nmap scan report for www.google.com (74.125.131.103)
Host is up (0.21s latency).
Hostname www.google.com resolves to 6 IPs. Only
   scanned 74.125.131.103
rDNS record for 74.125.131.103: vc-in-f103.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap scan report for www.facebook.com (31.13.69.160)
Host is up (0.18s latency).
rDNS record for 31.13.69.160: edge-star-shv-12-iad1.
   facebook.com
Not shown: 997 filtered ports
PORT    STATE  SERVICE
80/tcp  open   http
443/tcp open   https
843/tcp closed unknown

Nmap done: 2 IP addresses (2 hosts up) scanned in
   31.96 seconds
```

Outra opção é rastrear uma rede completa como na Listagem 9.35.

Listagem 9.35: Rastreando uma Sub-rede

```
vivas@zafu:~$ nmap 192.168.0.*

Starting Nmap 5.21 ( http://nmap.org ) at 2013-12-17
   15:49 BRST
Nmap scan report for 192.168.0.1
Host is up (0.048s latency).
```

```
Not shown: 998 closed ports
PORT   STATE SERVICE
53/tcp open  domain
80/tcp open  http

Nmap scan report for 192.168.0.101
Host is up (0.00034s latency).
All 1000 scanned ports on 192.168.0.101 are closed

Nmap scan report for 192.168.0.102
Host is up (0.0028s latency).
Not shown: 865 closed ports, 134 filtered ports
PORT      STATE SERVICE
62078/tcp open  iphone-sync

Nmap scan report for 192.168.0.104
Host is up (0.0082s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 256 IP addresses (4 hosts up) scanned in
   170.59 seconds
```

9.15 Comando route

O comando **route** possibilita a manipulação de rotas de roteamento. Se quiser verificar as rotas presentes em seu computador, basta utilizar a Listagem 9.36

Listagem 9.36: Visualizando a Tabela de Roteamento

```
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags
   Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG
   0      0        0 eno1
200.131.20.192  0.0.0.0         255.255.255.240 U
   0      0        0 eno1
```

9.16 Comando telnet

O comando **telnet** foi muito utilizado como protocolo de acesso remoto. Aos poucos foi substituído pelo comando **ssh** devido a problemas de segurança.

9.16.1 Acessando Servidor Web via Telnet

Na Listagem 9.37 apresenta o comando para conectar ao servidor desejado.

Listagem 9.37: Uso do Telnet

```
$ telnet www.vivas.eng.br 80
```

Você irá receber a seguinte resposta do servidor como na Listagem 9.38.

Listagem 9.38: Resposta do Servidor

```
Trying 208.115.217.250...
Connected to vivas.eng.br.
Escape character is '^]'.
```

Digite os comandos seguintes da Listagem 9.39 e termine pressionando duas vezes enter.

Listagem 9.39: Acessando com Telnet o Servidor Web

```
GET / HTTP/1.1
Host: vivas.eng.br
```

Como resposta, o servidor enviará informações do protocolo e enviará a página desejada como na Listagem 9.40. Retiramos a página do código para economia de espaço

Listagem 9.40: Resposta do Servidor

```
HTTP/1.1 200 OK
Server: nginx admin
Date: Tue, 17 Dec 2013 21:37:38 GMT
Content-Type: text/html
Content-Length: 7032
Connection: keep-alive
Vary: Accept-Encoding
Last-Modified: Mon, 16 Dec 2013 19:40:00 GMT
Accept-Ranges: bytes
X-Cache: HIT from Backend

Pagina

Connection closed by foreign host.
```

9.17 Acesso Remoto com ssh

O comando **ssh** permite o acesso remoto a um servidor. O primeiro passo é a instalação do pacote. A Listagem 9.41 apresenta o código para instalação do aplicativo.

Listagem 9.41: Instalando ssh

```
$ sudo apt-get install openssh-client
$ sudo apt-get install openssh-server
```

9.17.1 Acesso Remoto

Para acessar remotamente um servidor, basta você fazer o procedimento da Listagem 9.42, onde **vivas** é o usuário e o endereço IP do servidor é 192.168.0.1.

Listagem 9.42: Utilizando o ssh

```
$ ssh vivas@192.168.0.1
```

9.17.2 Rodando Aplicativos Gráficos Remotamente

Para rodar aplicativos gráficos remotamente via **ssh** você precisa alterar o arquivo de configuração do arquivo */etc/ssh/ssh_config*. Para isto, abra o arquivo com modo privilegiado e mude a seguinte linha do arquivo: **ForwardX11 no** para **ForwardX11 yes** como na Listagem 9.43.

Listagem 9.43: Alterando arquivo de Configuração do SSH

```
    ForwardX11 yes
```

Depois reinicialize o servidor **ssh** conforme Listagem 9.44.

Listagem 9.44: Rodando Aplicativos Gráficos Remotamente

```
sudo /etc/init.d/ssh restart
Rather than invoking init scripts through /etc/init.d,
    use the service(8)
utility, e.g. service ssh restart

Since the script you are attempting to invoke has been
    converted to an
Upstart job, you may also use the stop(8) and then
   start(8) utilities,
e.g. stop ssh ; start ssh. The restart(8) utility is
   also available.
```

```
ssh stop/waiting
ssh start/running, process 2532
```

Para logar exportando a parte gráfica utilizando a opção `-X` conforme Listagem 9.45.

Listagem 9.45: Logando com ssh -X

```
$ ssh -X vivas@192.168.0.104
```

Depois de logar digite o nome do aplicativo desejado seguido de `&` conforme Listagem 9.46.

Listagem 9.46: Abrindo Firefox Remotamente

```
$ firefox &
```

9.18 Copiando Arquivos com scp

O ssh permite também que você copie um arquivo de um computador remoto para outro computador remoto. Neste exemplo vamos copiar o arquivo *teste.txt* (que está no diretório */home/vivas*) que está no computador 192.168.0.104 para o meu computador para o diretório */users/alessandrovivas*. Repare que você vai digitar a senha do seu computador remoto e não do computador que você está logado . Para fazer esta tarefa utilize o comando **scp** e o código está na Listagem 9.47.

Listagem 9.47: Copiando Arquivo em Servidor Remoto

```
$ scp vivas@192.168.0.104:/home/vivas/teste.txt /users
   /alessandrovivas

vivas@192.168.0.104's password:
teste.txt 100% 438 0.4KB/s 00:00
```

9.19 Copiando um Diretório em um Servidor Remoto

Imagine que você criou um diretório em um servidor remoto. Vamos supor que o diretório tem o nome de ubuntu (*/home/vivas/ubuntu*). Para copiar o diretório inteiro, todos os arquivos, e criar a mesma estrutura no seu computador basta usar o comando scp. O código está apresentado na Listagem 9.48 e vamos utilizar o comando **scp**.

Listagem 9.48: Copiando um Diretório de um Servidor Remoto

```
$ scp -r vivas@192.168.0.104:/home/vivas/ubuntu /users
   /alessandrovivas/
vivas@192.168.0.104's password:
arquivo2      100%    0     0.0KB/s   00:00
arquivo1      100%    0     0.0KB/s   00:00
arquivo3      100%    0     0.0KB/s   00:00
```

9.20 Comando tcpdump

O comando **tcpdump** é utilizado para obter informações de suas conexões de rede e pode atuar como um sniffer. Para listar as interfaces de rede que ele pode escutar utilize a Listagem 9.50. Em algumas distribuições é necessário realizar o processo de instalação como na Listagem 9.49.

Listagem 9.49: Instalação do tcpdump

```
root@masamune:~# apt install tcpdump
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os NOVOS pacotes a seguir serão instalados:
  tcpdump
0 pacotes atualizados, 1 pacotes novos instalados, 0 a
    serem removidos e 1 não atualizados.
É preciso baixar 417 kB de arquivos.
Depois desta operação, 1.221 kB adicionais de espaço
   em disco serão usados.
Obter:1 http://ftp.br.debian.org/debian buster/main
   amd64 tcpdump amd64 4.9.2-3 [417 kB]
Baixados 417 kB em 35s (11,8 kB/s)
A seleccionar pacote anteriormente não seleccionado
   tcpdump.
(Lendo banco de dados ... 317664 ficheiros e directó
   rios actualmente instalados.)
A preparar para desempacotar .../tcpdump_4.9.2-3_amd64
   .deb ...
A descompactar tcpdump (4.9.2-3) ...
Configurando tcpdump (4.9.2-3) ...
A processar 'triggers' para man-db (2.8.5-2) ...
```

Listagem 9.50: Interfaces que podem ser utilizadas com tcpdump

```
# tcpdump -D
1.eno1 [Up, Running]
```

```
2.any (Pseudo-device that captures on all interfaces)
   [Up, Running]
3.lo [Up, Running, Loopback]
4.nflog (Linux netfilter log (NFLOG) interface)
5.nfqueue (Linux netfilter queue (NFQUEUE) interface)
6.usbmon1 (USB bus number 1)
7.usbmon2 (USB bus number 2)
```

Para realizar a captura de pacotes da interface de rede utilize a Listagem 9.51. Para sair digite <Ctrl+C>.

Listagem 9.51: Capturando Pacotes da Interface de Rede

```
root@masamune:~# tcpdump -i eno1
tcpdump: verbose output suppressed, use -v or -vv for
   full protocol decode
listening on eno1, link-type EN10MB (Ethernet),
   capture size 262144 bytes
17:19:09.232623 IP masamune.39468 > 162.248.19.147.
   https: Flags [.], ack 2763059533, win 312, options
   [nop,nop,TS val 1769479963 ecr 1088476037], length
   0
17:19:09.232639 IP masamune.56294 > 199.187.193.166.
   https: Flags [.], ack 3439314639, win 37960, length
    0
17:19:09.232687 IP masamune.39702 > a95-100-45-198.
   deploy.static.akamaitechnologies.com.https: Flags
   [.], ack 1221273869, win 383, options [nop,nop,TS
   val 26119193 ecr 2062682858], length 0
17:19:09.233557 IP masamune.40733 > 10.253.16.5.domain
   : 7930+ PTR? 147.19.248.162.in-addr.arpa. (45)
....
```

Para capturar pacotes de um protocolo específico, utilize o nome do protocolo após o comando, como na Listagem 9.52.

Listagem 9.52: Filtrando Pacotes pelo Protocolo

```
# tcpdump -i eno1 tcp
tcpdump: verbose output suppressed, use -v or -vv for
   full protocol decode
listening on eno1, link-type EN10MB (Ethernet),
   capture size 262144 bytes
17:20:28.882236 IP masamune.44400 > 153.232.73.34.bc.
   googleusercontent.com.https: Flags [P.], seq
   2581032862:2581032908, ack 3783587343, win 410,
```

```
   options [nop,nop,TS val 4082030995 ecr 1516370543],
    length 46
17:20:28.882320 IP masamune.33634 > gru06s28-in-f3.1
   e100.net.https: Flags [P.], seq
   555976556:555976595, ack 628241118, win 590,
   options [nop,nop,TS val 3795774733 ecr 1823276479],
    length 39

...

# tcpdump -i eno1 icmp
tcpdump: verbose output suppressed, use -v or -vv for
   full protocol decode
listening on eno1, link-type EN10MB (Ethernet),
   capture size 262144 bytes
17:21:40.258543 IP masamune > 10.253.16.5: ICMP
   masamune udp port 36954 unreachable, length 80
17:21:40.260050 IP masamune > 10.253.16.5: ICMP
   masamune udp port 36954 unreachable, length 80
17:21:51.879892 IP masamune > 10.253.16.5: ICMP
   masamune udp port 43985 unreachable, length 80
^C
3 packets captured
3 packets received by filter
0 packets dropped by kernel
```

9.21 Navegando no Terminal

O aplicativo **lynx** permite a navegação na Internet no terminal. Para utilizar este aplicativo primeiro realize a instalação como na Listagem 9.53.

Listagem 9.53: Instalação do lynx

```
$ sudo apt install lynx
Password:
```

Depois é só utilizar através do comando da Listagem 9.54.

Listagem 9.54: Utilizando o lynx

```
$lynx www.google.com
```

A Figura 9.1 apresenta o resultado do comando.

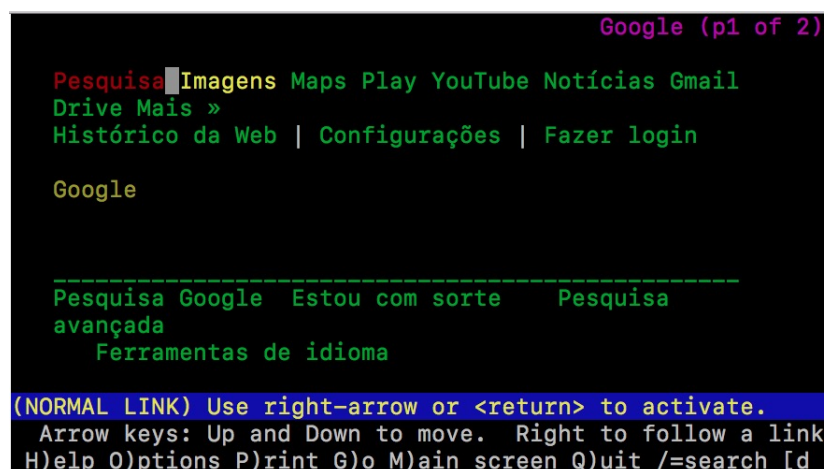



Figura 9.1: Interface do lynx

9.22 Baixando Sites com wget

Para baixar um site inteiro podemos utilizar o comando **wget** . A sintaxe é apresentada na Listagem 9.55 e para interromper digite <Ctrl+C>.

Listagem 9.55: Baixando Sites com wget

```
$ wget --recursive www.vivas.eng.br
--2013-12-18 07:49:33--  http://www.vivas.eng.br/
Resolvendo www.vivas.eng.br (www.vivas.eng.br)...
   208.115.217.250
Conectando-se a www.vivas.eng.br (www.vivas.eng.br)
   |208.115.217.250|:80... conectado.
A requisicaoo HTTP foi enviada, aguardando resposta...
    200 OK
Tamanho: 7032 (6,9K) [text/html]
Salvando em: www.vivas.eng.br/index.html

100%[=============================================================
    7.032       11,3K/s   em 0,6s

2013-12-18 07:49:35 (11,3 KB/s) - www.vivas.eng.br/
   index.html salvo [7032/7032]
```

Capítulo 10

Gerenciamento de Pacotes

Sumário

10.1 Repositório

O Linux utiliza um repositório de pacotes e todas as operações de instalação e remoção podem ser feitas utilizando comandos. Um repositório é um servidor onde os pacotes estão armazenados. Iremos tratar aqui do gerenciamento de pacotes das distribuições da família Debian (Ubuntu, Linux Mint, TAILS, Knoppix, dentre outras). Para instalar você digita o comando e o nome do pacote, nada além disto. Sua máquina entra em contato com o servidor, faz o download do pacote e depois instala automaticamente o software.

10.2 Atualização de Pacotes

Para atualizar a listagem dos pacotes disponíveis utilizamos o comando **apt**. A Listagem 10.1 ilustra o comando. No Debian, você precisa entrar como usuário root e em outras distribuições basta digitar o comando sudo antes do comando.

Listagem 10.1: Atualização da Lista de Pacotes Disponíveis

```
$ apt update
Obter:1 http://security.debian.org/debian-security
   buster/updates InRelease [39,1 kB]
Obter:2 http://security.debian.org/debian-security
   buster/updates/main Sources [65,2 kB]
Obter:3 http://security.debian.org/debian-security
   buster/updates/main amd64 Packages [86,8 kB]
Obter:4 http://security.debian.org/debian-security
   buster/updates/main Translation-en [53,2 kB]
Atingido:5 http://repository.spotify.com stable
   InRelease
Atingido:6 http://ftp.br.debian.org/debian buster
   InRelease
Atingido:7 http://ftp.br.debian.org/debian buster-
   updates InRelease
Ign:9 http://dl.google.com/linux/chrome/deb stable
   InRelease
Atingido:8 http://cdn-fastly.deb.debian.org/debian
   buster InRelease
Atingido:10 http://dl.google.com/linux/chrome/deb
   stable Release
Baixados 244 kB em 2s (151 kB/s)
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
```

```
4 packages can be upgraded. Run 'apt list --upgradable
   ' to see them.
```

10.3 Atualizando a Distribuição

A medida que o tempo vai passando novas versões de software são disponibilizadas. Diversas atualizações de segurança são realizadas em cada semana. Para manter sua distribuição atualizada, você precisa utilizar o comando upgrade como na Listagem 10.2.

Listagem 10.2: Atualização de Pacotes

```
$ apt upgrade
```

A Listagem 10.3 apresenta o comando para consultar quais pacotes podem ser atualizados.

Listagem 10.3: Lista de Pacotes para Upgrade

```
# apt list --upgradable
Listing... Pronto
gir1.2-ibus-1.0/stable 1.5.19-4+deb10u1 amd64 [
   upgradable from: 1.5.19-4]
google-chrome-stable/stable 77.0.3865.90-1 amd64 [
   upgradable from: 76.0.3809.132-1]
libibus-1.0-5/stable 1.5.19-4+deb10u1 amd64 [
   upgradable from: 1.5.19-4]
tzdata/stable-updates 2019c-0+deb10u1 all [upgradable
   from: 2019b-0+deb10u1]
```

10.4 Instalando Softwares

Para instalar um novo pacote você precisa saber o nome do software e utilizar o comando install como na Listagem 10.4.

Listagem 10.4: Instalação do Pacote vim

```
$ apt install vim
```

10.5 Removendo Pacotes

Para remover um pacote específico você precisa saber o nome do software e utilizar o comando remove como na Listagem 10.5.

Listagem 10.5: Removendo o Pacote vim

```
$ apt remove vim
```

10.6 Instalando Software no Fedora

Para instalar um novo pacote no Fedora você precisa utilizar o comando **dnf**. A Listagem 10.6 apresenta o procedimento para instalação do pacote **lshw**.

Listagem 10.6: Instalação do Pacote lshw no Fedora

```
dnf install lshw
```

Capítulo 11

Comandos Úteis

Sumário

11.1 Comando unit

O comando **unit** é utilizado para realizar conversões entre unidades. A Listagem 11.1 mostra como converter de metros para quilômetros.

Listagem 11.1: Exemplos de utilização do comando **units**

```
$ units 5inches cm
        * 12.7
        / 0.078740157
$ units 1mile km
        * 1.609344
        / 0.62137119
$ units
Currency exchange rates from www.timegenie.com on
   2014-04-02
2866 units, 109 prefixes, 79 nonlinear units

You have: 10 ounces
You want: grams
        * 283.49523
        / 0.0035273962
```

11.2 Comando yes

O comando **yes** é utilizado para responder automaticamente a perguntas em scripts. A Listagem 11.2 mostra um exemplo de utilização. Para terminar digite <Ctrl+C>.

Listagem 11.2: Exemplo de utilização do comando **yes** para responder automaticamente a perguntas com ‘yes’

```
$ touch file1 file2 file3 && yes | rm -i file1 file2
   file3
```

Ele pode ser utilizado para imprimir mensagens repetidas indefinidamente em seu terminal como na Listagem 11.3.

Listagem 11.3: Imprimindo uma mensagem indefinidamente no terminal utilizando o **yes**

```
$ yes ‘hoje é sexta!’
hoje é sexta!
hoje é sexta!
hoje é sexta!
hoje é sexta!
```

```
hoje é sexta!^C
```

Em um script para compilar texto em Latex utilizo o comando **yes** para responder **r** quando ocorre erro no processamento. A Listagem 11.4 apresenta o exemplo do uso do comando.

Listagem 11.4: Utilizando o Comando yes para Processamento de Latex

```
#!/bin/bash
yes r | pdflatex artigo.tex
bibtex biblio
makeindex artigo
```

Outra utilização do comando **yes** é criar uma forma rápida de apagar um diretório de forma recursiva sem precisar responder **yes** repetidamente.

Listagem 11.5: Utilizando o Comando yes para apagar um diretório grande

```
yes | rm -r diretorio_grande
```

11.3 Comando bc

O comando **bc** é utilizado como uma calculadora na linha de comando. Ele permite fazer operações aritmética, utilizar variáveis, incrementar e decrementar, utilizar funções matemáticas, etc.

O **bc** processa as operações descritas em um arquivo ou enviados pela entrada padrão. Vejamos alguns exemplos simples de utilização.

Listagem 11.6: Utilizando o comando bc

```
$ echo "7+5" | bc
12
$ echo "x=10;y=x^2-1;y" | bc
99
$ echo "x=10;x%3" | bc
1
```

Vamos utilizar o **bc** agora para somar os valores em um arquivo. Em vez de utilizar um arquivo propriamente dito, iremos utilizar o resultado gerado pelo comando **seq**.

Listagem 11.7: Utilizando o comando bc para somar valores em um arquivo

```
$ paste -sd+ <(seq 1 5)
1+2+3+4+5
$ paste -sd+ <(seq 1 5) | bc
15
```

Podemos também realizar cálculos utilizando funcões matemáticas. Para tanto é necessário utilizar o parâmetro **-l**. No exemplo a seguir vamos armazenar o valor de π em uma varável chamada **pi**. Iremos utilizar a seguinte relação: $\pi = 4\arctan(1)$. Em seguida utilizaremos o valor de π para calcular o seno de $3\pi/2$.

Listagem 11.8: Utilizando o comando bc para somar valores em um arquivo

```
$ pi=`echo "4*a(1)" | bc -l`
$ echo $pi
3.14159265358979323844
$ echo "s(3*$pi/2)" | bc -l
-.99999999999999999999
```

No exemplo a seguir vamos ilustrar algumas formas de somar os tamanhos dos arquivos (no caso apenas os arquivos .png) no diretório corrente.

Listagem 11.9: Utilizando o comando bc para somar os tamanhos dos arquivos

```
$ ls -la *.png | sed 's/\s\s*/ /g' | cut -d' ' -f5 |
   paste -sd+ | bc
2563681
$ ls -la *.png | awk '{print $5}' | paste -sd+ | bc
2563681
$ ls -la *.png | awk '{S+=$5} END{print S}'
2563681
```

Capítulo 12

Comandos Divertidos

Sumário

12.1 Comando cowsay

O comando **cowsay** funciona apenas no Linux. Para utilizar você precisa realizar a instalação do mesmo, Listagem 12.1 para Debian e Ubuntu. Para Fedora utilize o comando da Listagem 12.2.

Listagem 12.1: Instalação do Comando cowsay no Debian/Ubuntu

```
$ sudo apt install cowsay
Password:
```

Listagem 12.2: Instalação do Comando cowsay no Fedora

```
[root@musashi ~]# dnf install cowsay
```

Após a instalação o comando está pronto para o uso, Listagem 12.3.

Listagem 12.3: Comando cowsay

```
[avivas@musashi ~]$ cowsay "Eu amo Linux"
 ______________
< Eu amo Linux >
 --------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

12.2 Comando xcowsay

O comando **xcowsay** funciona apenas no Linux. Para utilizar você precisa realizar a instalação do mesmo, Listagem 12.4.

Listagem 12.4: Comando cowsay

```
# para Ubuntu e Debian
$ sudo apt-get install xcowsay
Password:
# para Fedora
[root@musashi ~]$ dnf install xcowsay
```

Após a instalação o comando está pronto para o uso, Listagem 12.5.

Listagem 12.5: Comando cowsay

```
$xcowsay Eu amo Linux
```

A Figura 12.1 apresenta o resultado.

Figura 12.1: Comando xcowsay

12.3 Comando cowthink

O comando **cowthink** ou **xcowthink** altera apenas o balão de diálogo do comando **cowsay** (Listagem 12.6).

Listagem 12.6: Comando cowthink

```
$ cowthink "Oi"
 ----
( Oi )
 ----
        o   ^__^
         o  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

12.4 Comando fortune

O comando **fortune** envia frases aleatórias no terminal. A Listagem 12.7 explica como utilizar o comando.

Listagem 12.7: Comando fortune

```
[avivas@musashi ~]$ fortune
I think...  I think it's in my basement... Let me go
   upstairs and check.
                -- Escher
```

12.5 Comando xcowfortune

O comando **xcowfortune** comando utiliza o **xcowsay** e o **fortune** em conjunto. Para utilizar digite o comando da Listagem 12.8.

Listagem 12.8: Comando xcowfortune

```
$ xcowfortune
```

A Figura 12.2 apresenta o resultado.

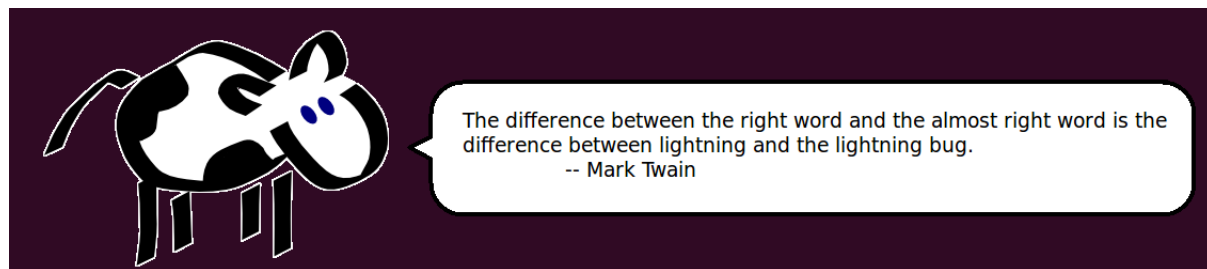



Figura 12.2: Comando xcowfortune

12.6 Comando sl

Outro comando interessante é o **sl** . O procedimento de instalação é apresentado na Listagem 12.9. Para executá-lo utilize a Listagem 12.10 e o resultado é apresentado na Figura 12.3.

Listagem 12.9: Comando **sl**

```
# Ubuntu ou Debian
$ sudo apt-get install sl
# Fedora
[root@musashi ~]# dnf install sl
```

Listagem 12.10: Comando **sl**

```
$ sl
```

12.7 Comando xeyes

O comando **xeyes** apresenta dois olhos que acompanham o posicionamento do mouse. Para utilizá-lo, digite apenas **xeyes** no terminal como apresentado na Figura 12.4. Para instalar o aplicativo utilize o comando da Listagem 12.11.

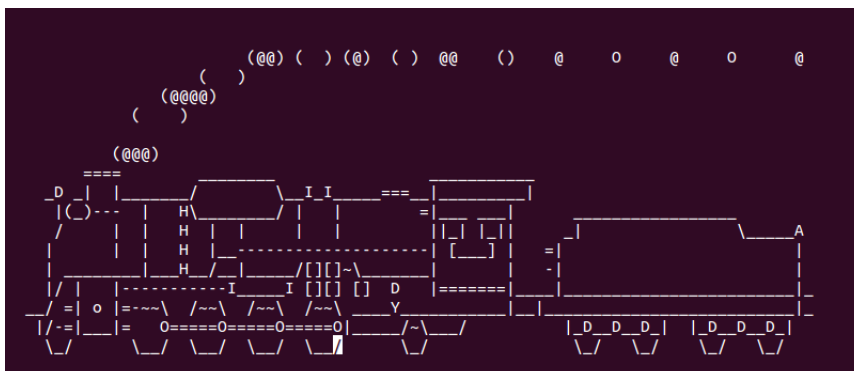



Figura 12.3: Comando sl

Listagem 12.11: Comando **xeyes**

```
# Ubuntu ou Debian
$ sudo apt-get install xeyes
# Fedora
[root@musashi ~]# dnf install xeyes
```

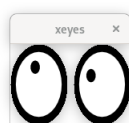



Figura 12.4: Comando xyes

12.8 Comando oneko

Oneko é um comando que cria um gatinho no seu desktop. O ponteiro de seu mouse vira um ratinho e o gatinho sempre fica tentando capturá-lo. A Listagem 12.12 mostra como instalar e como rodar o programa.

Listagem 12.12: Comando oneko

```
$ sudo apt-get install oneko
$ oneko
```

Se quiser mudar a imagem para um cachorro digite **$ oneko -dog** e para um tigre digite **$ oneko -tora**.

Capítulo 13

Comandos para Analisar o Desempenho do Linux

Sumário

13.1 Analisando Consumo de CPU com o Comando sar

O comando **sar** é utilizado para medir o consumo de CPU de qualquer computador. Para utilizar o comando instale o pacote sysstat como na Listagem 13.1.

Listagem 13.1: Instalando Pacote sysstat

```
$ sudo apt install sysstat
```

O comando **sar -u** precisa de dois argumentos: a) o número de segundos onde cada leitura é feita e b) a quantidade de relatórios que ele irá imprimir. A Listagem 13.2 apresenta o relatório de consumo de CPU medida a cada 1 segundo durante 10 medições.

Listagem 13.2: Analisando Desempenho da CPU com sar

```
[root@musashi ~]# sar -u 1 10
Linux 4.0.4-301.fc22.x86_64 (musashi.vivascorp)        18-09-2015      _x86_64_
          (4 CPU)

15:28:39      CPU     %user     %nice   %system   %iowait    %steal     %idle
15:28:40      all      0,00      0,00      0,00      3,52      0,00     96,48
15:28:41      all      0,00      0,00      0,25      0,25      0,00     99,50
15:28:42      all      0,00      0,00      0,00      0,00      0,00    100,00
15:28:43      all      0,00      0,00      0,00      0,00      0,00    100,00
15:28:44      all      0,00      0,00      0,25      0,25      0,00     99,50
15:28:45      all      0,00      0,00      0,00      0,75      0,00     99,25
15:28:46      all      0,00      0,00      0,00      0,00      0,00    100,00
15:28:47      all      0,00      0,00      0,00      0,00      0,00    100,00
15:28:48      all      0,00      0,00      0,25      0,00      0,00     99,75
15:28:49      all      0,00      0,00      0,00      0,00      0,00    100,00
Média:        all      0,00      0,00      0,08      0,48      0,00     99,45
```

Onde %usr é a quantidade de CPU utilizada pelo sistema com processos dos usuários , %sys é o percentual de processo consumido por processos do sistema, %idle é o percentual de CPU ocioso e %nice é o percentual de CPU consumidos por processos que tenham algum tipo de prioridade de escalonamento.

13.2 Analisando Desempenho de CPU com mpstat

O comando **mpstat** é utilizado para medir o desempenho de CPU de qualquer computador. É possível analisar como está cada núcleo de sua CPU. A Listagem 13.3 apresenta o relatório de desempenho de CPU para todos os núcleos.

Listagem 13.3: Analisando Desempenho de Todos os Núcleos com mpstat

```
[root@musashi ~]# mpstat -P ALL
Linux 4.0.4-301.fc22.x86_64 (musashi.vivascorp)        18-09-2015      _x86_64_
          (4 CPU)

15:31:59  CPU  %usr  %nice %sys  %iowait %irq  %soft %steal %guest %gnice %idle
15:31:59  all  0,09  0,01  0,04  0,52    0,00  0,00  0,00   0,00   0,00   99,35
15:31:59    0  0,07  0,00  0,04  0,70    0,00  0,00  0,00   0,00   0,00   99,18
15:31:59    1  0,06  0,01  0,04  0,58    0,00  0,00  0,00   0,00   0,00   99,31
15:31:59    2  0,10  0,01  0,03  0,40    0,00  0,00  0,00   0,00   0,00   99,46
15:31:59    3  0,14  0,00  0,04  0,39    0,00  0,00  0,00   0,00   0,00   99,43
```

13.3 Estatísticas de Entrada e Saída com iostat

O comando **iostat** é utilizado para fornecer estatísticas de entrada e saída e CPU de qualquer computador. A Listagem 13.4 apresenta o relatório das estatísticas de entrada e saída..

Listagem 13.4: Analisando Estatísticas de Entrada e Saída com iostat

```
$ iostat
Linux 4.15.0-58-generic (musashi)        09/27/2019       _x86_64_        (48 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          16.15    0.00    0.01    0.01    0.00   83.83

Device             tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
loop0             0.00         0.00         0.00       1307          0
loop1             0.00         0.00         0.00       1806          0
loop2             0.00         0.00         0.00       7092          0
loop3             0.01         0.01         0.00      13020          0
loop4             0.00         0.00         0.00          8          0
sda               3.81         1.25        52.86    2699144  114119092
```

13.4 Analisando a Memória com vmstat

O comando vmstat pode ser utilizado para analisar estatísticas sobre a memória. A Listagem 13.5 apresenta o resultado do comando.

Listagem 13.5: Analisando a Memória com vmstat

```
$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
100  0      0 94596584 586348 34707256    0    0     0     0    1    0  0 16  0 84
    0  0
```

13.5 Comando pidstat

O comando **pidstat** dá estatísticas sobre todos os processos que estão rodando em sua máquina. A Listagem 13.6 apresenta o resultado do comando.

Listagem 13.6: Comando pidstat

```
[root@musashi ~]# pidstat
Linux 4.0.4-301.fc22.x86_64 (musashi.vivascorp)        18-09-2015       _x86_64_
          (4 CPU)

16:02:01      UID       PID    %usr %system  %guest    %CPU   CPU  Command
16:02:01        0         1    0,00    0,00    0,00    0,00     3  systemd
16:02:01        0         2    0,00    0,00    0,00    0,00     0  kthreadd
16:02:01        0         3    0,00    0,00    0,00    0,00     0  ksoftirqd/0
16:02:01        0         7    0,00    0,06    0,00    0,06     1  rcu_sched
16:02:01        0         9    0,00    0,02    0,00    0,02     1  rcuos/0
16:02:01        0        11    0,00    0,00    0,00    0,00     0  migration/0
16:02:01        0        12    0,00    0,00    0,00    0,00     0  watchdog/0
16:02:01        0        13    0,00    0,00    0,00    0,00     1  watchdog/1
16:02:01        0        14    0,00    0,00    0,00    0,00     1  migration/1
16:02:01        0        15    0,00    0,00    0,00    0,00     1  ksoftirqd/1
16:02:01        0        18    0,00    0,01    0,00    0,01     0  rcuos/1
16:02:01        0        20    0,00    0,00    0,00    0,00     2  watchdog/2
16:02:01        0        21    0,00    0,00    0,00    0,00     2  migration/2
16:02:01        0        22    0,00    0,00    0,00    0,00     2  ksoftirqd/2
16:02:01        0        25    0,00    0,02    0,00    0,02     3  rcuos/2
16:02:01        0        27    0,00    0,00    0,00    0,00     3  watchdog/3
```

```
16:02:01     0     28   0,00   0,00   0,00   0,00   3  migration/3
16:02:01     0     29   0,00   0,00   0,00   0,00   3  ksoftirqd/3
16:02:01     0     32   0,00   0,01   0,00   0,01   1  rcuos/3
16:02:01     0     52   0,00   0,00   0,00   0,00   2  fsnotify_mark
16:02:01     0    131   0,00   0,00   0,00   0,00   3  kauditd
16:02:01     0    319   0,00   0,00   0,00   0,00   2  kworker/2:1H
16:02:01     0    320   0,00   0,00   0,00   0,00   3  kworker/3:1H
16:02:01     0    322   0,00   0,00   0,00   0,00   1  kworker/1:1H
16:02:01     0    328   0,00   0,00   0,00   0,00   0  kworker/0:1H
16:02:01     0    420   0,00   0,00   0,00   0,00   1  jbd2/dm-1-8
16:02:01     0    506   0,00   0,01   0,00   0,01   1  systemd-
   journal
16:02:01     0    525   0,00   0,00   0,00   0,00   0  lvmetad
16:02:01     0    550   0,00   0,00   0,00   0,00   2  systemd-udevd
16:02:01     0    580   0,00   0,00   0,00   0,00   1  ips-adjust
16:02:01     0    581   0,00   0,01   0,00   0,01   1  ips-monitor
16:02:01     0    687   0,00   0,00   0,00   0,00   0  jbd2/dm-2-8
16:02:01     0    708   0,00   0,00   0,00   0,00   1  auditd
16:02:01     0    717   0,00   0,00   0,00   0,00   1  alsactl
16:02:01     0    719   0,00   0,00   0,00   0,00   3  firewalld
16:02:01     0    720   0,00   0,00   0,00   0,00   0  systemd-logind
16:02:01    81    721   0,01   0,00   0,00   0,01   1  dbus-daemon
16:02:01     0    727   0,00   0,00   0,00   0,00   3  audispd
16:02:01     0    729   0,00   0,00   0,00   0,00   3  sedispatch
16:02:01    70    734   0,00   0,00   0,00   0,00   0  avahi-daemon
16:02:01   172    735   0,00   0,00   0,00   0,00   3  rtkit-daemon
16:02:01     0    738   0,00   0,00   0,00   0,00   1  accounts-
   daemon
16:02:01     0    739   0,00   0,00   0,00   0,00   0  abrtd
16:02:01   991    747   0,00   0,00   0,00   0,00   3  chronyd
16:02:01     0    757   0,00   0,00   0,00   0,00   1  gssproxy
16:02:01     0    769   0,00   0,00   0,00   0,00   3  abrt-watch-log
16:02:01     0    771   0,00   0,00   0,00   0,00   3  abrt-dump-
   journ
16:02:01   995    782   0,00   0,00   0,00   0,00   3  polkitd
16:02:01     0    839   0,01   0,01   0,00   0,01   2  NetworkManager
16:02:01     0    904   0,00   0,01   0,00   0,01   3  kworker/u8:2
16:02:01     0    916   0,00   0,00   0,00   0,00   2  libvirtd
16:02:01     0    989   0,00   0,00   0,00   0,00   1  crond
16:02:01     0    990   0,00   0,00   0,00   0,00   1  gdm
16:02:01     0    992   0,00   0,00   0,00   0,00   0  atd
16:02:01     0   1037   0,00   0,00   0,00   0,00   3  wpa_supplicant
16:02:01     0   1038   0,00   0,00   0,00   0,00   2  kworker/2:1
16:02:01     0   1344   0,00   0,00   0,00   0,00   0  gdm-session-
   wor
16:02:01    42   1356   0,00   0,00   0,00   0,00   1  Xorg
16:02:01    42   1385   0,00   0,00   0,00   0,00   0  dbus-daemon
16:02:01    42   1388   0,00   0,00   0,00   0,00   1  gnome-session
16:02:01    42   1391   0,00   0,00   0,00   0,00   0  at-spi-bus-
   laun
16:02:01    42   1394   0,00   0,00   0,00   0,00   1  gvfsd
16:02:01    42   1404   0,00   0,00   0,00   0,00   2  at-spi2-
   registr
16:02:01    42   1422   0,00   0,00   0,00   0,00   2  gnome-settings
   -
16:02:01     0   1428   0,00   0,00   0,00   0,00   0  upowerd
16:02:01     0   1454   0,00   0,00   0,00   0,00   3  kworker/3:1
16:02:01    42   1455   0,03   0,00   0,00   0,03   0  gnome-shell
16:02:01   994   1457   0,00   0,00   0,00   0,00   3  colord
16:02:01    42   1470   0,00   0,00   0,00   0,00   0  pulseaudio
16:02:01    42   1519   0,00   0,00   0,00   0,00   3  ibus-daemon
16:02:01    42   1524   0,00   0,00   0,00   0,00   2  ibus-dconf
16:02:01    42   1526   0,00   0,00   0,00   0,00   1  ibus-x11
16:02:01     0   1532   0,00   0,00   0,00   0,00   1  dhclient
16:02:01     0   1536   0,12   0,02   0,00   0,14   0  packagekitd
16:02:01    42   1538   0,00   0,00   0,00   0,00   1  gvfs-udisks2-
   vo
16:02:01     0   1542   0,01   0,00   0,00   0,02   3  udisksd
16:02:01    42   1566   0,00   0,00   0,00   0,00   1  goa-daemon
16:02:01    42   1572   0,00   0,00   0,00   0,00   1  mission-
   control
16:02:01    42   1629   0,00   0,00   0,00   0,00   1  ibus-engine-
   sim
16:02:01     0   1662   0,00   0,00   0,00   0,00   0  kworker/0:1
16:02:01     0   1663   0,00   0,00   0,00   0,00   2  kworker/u8:1
16:02:01     0   1685   0,00   0,00   0,00   0,00   0  sshd
16:02:01  1000   1690   0,00   0,00   0,00   0,00   2  sshd
16:02:01  1000   1693   0,00   0,00   0,00   0,00   0  bash
16:02:01     0   1694   0,00   0,00   0,00   0,00   0  gdm-session-
   wor
16:02:01  1000   1707   0,00   0,00   0,00   0,00   1  systemd
16:02:01  1000   1732   0,00   0,00   0,00   0,00   3  gnome-keyring-
   d
16:02:01  1000   1766   0,00   0,00   0,00   0,01   3  Xorg
```

```
16:02:01    1000    1829    0,00    0,00    0,00    0,00    1   dbus-daemon
16:02:01    1000    1834    0,00    0,00    0,00    0,00    2   gnome-session
16:02:01    1000    1921    0,00    0,00    0,00    0,00    1   at-spi-bus-
     laun
16:02:01    1000    1924    0,00    0,00    0,00    0,00    0   gvfsd
16:02:01    1000    1928    0,00    0,00    0,00    0,00    2   gvfsd-fuse
16:02:01    1000    1938    0,00    0,00    0,00    0,00    0   dbus-daemon
16:02:01    1000    1943    0,00    0,00    0,00    0,00    0   at-spi2-
     registr
16:02:01    1000    1959    0,00    0,00    0,00    0,00    1   su
16:02:01       0    1967    0,00    0,00    0,00    0,00    1   kworker/1:1
16:02:01       0    1970    0,00    0,00    0,00    0,00    0   bash
16:02:01    1000    1971    0,00    0,00    0,00    0,00    1   gnome-settings
     -
16:02:01    1000    1976    0,00    0,00    0,00    0,00    3   pulseaudio
16:02:01    1000    2037    0,05    0,00    0,00    0,05    2   gnome-shell
16:02:01       0    2041    0,00    0,00    0,00    0,00    2   cupsd
16:02:01    1000    2051    0,00    0,00    0,00    0,00    3   gsd-printer
16:02:01    1000    2059    0,00    0,00    0,00    0,00    3   gnome-shell-
     cal
16:02:01    1000    2060    0,00    0,00    0,00    0,00    0   ibus-daemon
16:02:01    1000    2065    0,00    0,00    0,00    0,00    2   ibus-dconf
16:02:01    1000    2067    0,00    0,00    0,00    0,00    2   ibus-x11
16:02:01    1000    2076    0,00    0,00    0,00    0,00    1   evolution-
     sourc
16:02:01       0    2091    0,00    0,00    0,00    0,00    2   kworker/2:2
16:02:01    1000    2097    0,00    0,00    0,00    0,00    3   mission-
     control
16:02:01    1000    2102    0,00    0,00    0,00    0,00    1   goa-daemon
16:02:01    1000    2116    0,00    0,00    0,00    0,00    2   gvfs-udisks2-
     vo
16:02:01       0    2183    0,00    0,00    0,00    0,00    0   pidstat
16:02:01    1000    2219    0,00    0,00    0,00    0,00    0   tracker-store
16:02:01    1000    2220    0,00    0,00    0,00    0,00    0   abrt-applet
16:02:01    1000    2221    0,00    0,00    0,00    0,00    1   tracker-miner-
     f
16:02:01    1000    2227    0,00    0,00    0,00    0,00    1   evolution-
     alarm
16:02:01    1000    2239    0,00    0,00    0,00    0,00    1   gnome-software
16:02:01    1000    2255    0,00    0,00    0,00    0,00    0   ibus-engine-
     sim
16:02:01    1000    2302    0,00    0,00    0,00    0,00    1   seapplet
16:02:01    1000    2316    0,00    0,00    0,00    0,00    0   tracker-miner-
     u
16:02:01    1000    2318    0,00    0,00    0,00    0,00    1   evolution-
     calen
16:02:01    1000    2324    0,00    0,00    0,00    0,00    3   tracker-
     extract
16:02:01    1000    2327    0,00    0,00    0,00    0,00    0   tracker-miner-
     a
16:02:01    1000    2346    0,00    0,00    0,00    0,00    1   evolution-
     calen
16:02:01    1000    2364    0,00    0,00    0,00    0,00    2   evolution-
     addre
16:02:01    1000    2367    0,00    0,00    0,00    0,00    2   evolution-
     calen
16:02:01    1000    2391    0,00    0,00    0,00    0,00    3   evolution-
     addre
16:02:01    1000    2425    0,00    0,00    0,00    0,00    2   gnome-terminal
     -
16:02:01    1000    2429    0,00    0,00    0,00    0,00    3   bash
16:02:01    1000    2559    0,05    0,01    0,00    0,06    1   firefox
16:02:01       0    2814    0,00    0,00    0,00    0,00    2   sshd
16:02:01       0    3456    0,00    0,00    0,00    0,00    3   usb-storage
```

13.6 Comando top

O comando **top** é utilizado para fornecer estatísticas de uso de CPU, memória e diversas estatísticas de uso. A Figura 13.1 apresenta o relatório das estatísticas de uso de CPU. Para sair digite **q**. Para apresentar relatório de todas as CPUS digite 1.

```
top - 08:14:52 up 24 days, 23:43,  1 user,  load average: 100.02, 100.19, 10
Tasks: 580 total,  19 running, 298 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
KiB Mem : 13200879+total, 94595216 free,  2119600 used, 35293976 buff/cache
KiB Swap:  8388604 total,  8388604 free,        0 used. 12881764+avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
13321 jonathan  20   0  604480   8184   4460 R 201.0  0.0   2601:31 VRPSPD_+
13002 jonathan  20   0  669380   8144   4604 S 199.7  0.0   2632:03 VRPSPD_+
12947 jonathan  20   0  604616   8376   4628 R 199.4  0.0   2626:43 VRPSPD_+
13193 jonathan  20   0  604496   8272   4556 R 199.0  0.0   2619:44 VRPSPD_+
13258 jonathan  20   0  604580   8204   4612 R 198.7  0.0   2606:15 VRPSPD_+
13070 jonathan  20   0  604492   8316   4576 R 191.3  0.0   2427:22 VRPSPD_+
13446 jonathan  20   0  669372   7932   4476 S 178.5  0.0   2412:14 VRPSPD_+
12885 jonathan  20   0  604492   8224   4544 R 174.4  0.0   2448:27 VRPSPD_+
13386 jonathan  20   0  604692   8308   4596 R 171.8  0.0   2417:13 VRPSPD_+
13134 jonathan  20   0  604492   8324   4572 R 168.9  0.0   2433:19 VRPSPD_+
14349 jonathan  20   0  538308   8476   4680 S 132.7  0.0   1182:09 VRPSPD_+
14083 jonathan  20   0  473688   7324   4636 R 109.3  0.0   1187:15 VRPSPD_+
```

Figura 13.1: Uso do top para Obter Estatísticas de CPU

Capítulo 14

Verificando Configuração de Hardware e Software

Sumário

14.1 Visualizando Informações sobre a Versão do Kernel

Para verificar informações sobre a versão da sua Distribuição Linux utilize o comando **uname** como na Listagem 14.1. Vamos dar o exemplo deste mesmo comando em duas máquinas diferentes. A primeira máquina roda a distribuição Debian e o segundo comando roda Ubuntu.

Listagem 14.1: Versão do Kernel

```
vivas@masamune:~$ uname -a
Linux masamune 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+
   deb10u2 (2019-08-08) x86_64 GNU/Linux
vivas@musashi:~$ uname -a
Linux musashi 4.15.0-52-generic #56-Ubuntu SMP Tue Jun
    4 22:49:08 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

O comando apresenta o nome da máquina (masamune), o nome da sua distribuição (Debian), a versão da distribuição (4.19.37-5+deb10u2), a data (2019-08-08) e a arquitetura de sua máquina (x86_64). A segunda máquina tem nome de musashi, Ubuntu e possui uma versão mais antiga do kernel.

Este comando mostra uma versão completa dos dados, mas você pode decompor seu comando utilizando opções descritas na Tabela 14.1. Este comando é muito útil para verificação da versão corrente do kernel e a plataforma de sua máquina. Este conjunto de instruções x86_64 é compatível com os processadores da Intel e da AMD. Para saber mais sobre este assunto verifique a referência AMD64 da Wikipédia [12].

Comando	Descrição
uname -s	Nome do seu sistema operacional
uname -n	Nome da sua máquina (hostname)
uname -r	Release do Kernel
uname -v	Versão do Kernel

Tabela 14.1: Varições do comando uname

A Listagem 14.2 apresenta o resultado destes comandos.

Listagem 14.2: Versão do Kernel

```
vivas@masamune:~$ uname -s
Linux
vivas@masamune:~$ uname -n
masamune
vivas@masamune:~$ uname -r
4.19.0-5-amd64
```

```
vivas@masamune:~$ uname -v
#1 SMP Debian 4.19.37-5+deb10u2 (2019-08-08)
```

A string 4.19.37-5+deb10u2 apresenta o seguinte significado:

- 4 - Versão do Kernel
- 19 - Número da Versão Principal (major release)
- 37 - Revisão Menor (minor revision)
- 5 - Número de Correção (patch number)
- deb10u2 - string específica da distribuição

14.2 Verificando sua Distribuição

Uma distribuição Linux pode ser definida como um sistema operacional constituído do kernel do Linux e um conjunto de softwares. Mais informações sobre este conceito pode ser encontrado na referência Distribuição Linux [13].

Para verificar informações sobre sua distribuição, utilize o comando **head** como na Listagem 14.3.

Listagem 14.3: Verificando sua Distribuição

```
vivas@musashi:~$ head -n1 /etc/issue
Ubuntu 18.04.2 LTS \n \l
```

14.3 Visualizando Informações sobre seus Processadores

A Unidade Central de Processamento (CPU) nada mais é do que o processador utilizado em seu computador [15]. O comando **lscpu** é utilizado para listar todas as informações sobre os seus processadores. A Listagem 14.4 apresenta o resultado do comando em um Servidor da Dell R900.

Listagem 14.4: Verificando Informações sobre a CPU

```
vivas@musashi:~$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              48
On-line CPU(s) list: 0-47
Thread(s) per core:  2
Core(s) per socket:  6
```

```
Socket(s):             4
NUMA node(s):          4
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 46
Model name:            Intel(R) Xeon(R) CPU
   E7530  @ 1.87GHz
Stepping:              6
CPU MHz:               1995.296
BogoMIPS:              3724.08
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              12288K
NUMA node0 CPU(s):     0,4,8,12,16,20,24,28,32,36,40,44
NUMA node1 CPU(s):     1,5,9,13,17,21,25,29,33,37,41,45
NUMA node2 CPU(s):     2,6,10,14,18,22,26,30,34,38,42,46
NUMA node3 CPU(s):     3,7,11,15,19,23,27,31,35,39,43,47
Flags:                 fpu vme de pse tsc msr pae mce
   cx8 apic sep mtrr pge mca cmov pat pse36 clflush
   dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
   rdtscp lm constant_tsc arch_perfmon pebs bts
   rep_good nopl xtopology nonstop_tsc cpuid
   aperfmperf pni dtes64 monitor ds_cpl vmx est tm2
   ssse3 cx16 xtpr pdcm dca sse4_1 sse4_2 x2apic
   popcnt lahf_lm pti ssbd ibrs ibpb stibp tpr_shadow
   vnmi flexpriority ept vpid dtherm ida flush_l1d
```

Outra maneira é consultar o diretório */proc* como na Listagem 14.5. O comando grep abre o arquivo cpuinfo do diretório proc e procura a ocorrência "model name". Podemos verificar neste exemplo que a máquina possui diversos processadores. Neste mesmo arquivo existem informações detalhadas sobre as CPUs.

Listagem 14.5: Verificando a CPU

```
vivas@musashi:~$ grep "model name" /proc/cpuinfo
model name      : Intel(R) Xeon(R) CPU           E7530  @
   1.87GHz
...
model name      : Intel(R) Xeon(R) CPU           E7530  @
   1.87GHz
```

14.4 Visualizando Informações sobre os Dispositivos USB

O comando **lsusb** é utilizado para listar todas as informações sobre as conexões USB. A Listagem 14.6 apresenta o resultado do comando. Podemos verificar que neste computador estão ligados uma impressora Brother e um mouse Logitech.

Listagem 14.6: Verificando a CPU

```
vivas@masamune:~$ lsusb
Bus 002 Device 005: ID 04f9:003b Brother Industries, Ltd
Bus 002 Device 004: ID 046d:c063 Logitech, Inc. DELL Laser
    Mouse
Bus 002 Device 003: ID 413c:2106 Dell Computer Corp. Dell
    QuietKey Keyboard
Bus 002 Device 002: ID 8087:0020 Intel Corp. Integrated Rate
     Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root
    hub
Bus 001 Device 002: ID 8087:0020 Intel Corp. Integrated Rate
     Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root
    hub
```

14.5 Listando Todos os Dispositivos PCI

PCI (Peripheral Component Interconnect — Interconector de Componentes Periféricos) é um barramento para conectar periféricos em computadores [14]. O comando **lspci** é utilizado para listar todas as informações sobre os dispositivos PCI. A Listagem 14.7 apresenta o resultado do comando. Optamos por apresentar apenas parte da listagem devido ao excesso de informações apresentadas.

Listagem 14.7: Listando Dispositivos PCI

```
vivas@musashi:~$ lspci
00:00.0 Host bridge: Intel Corporation 5520/5500/X58 I/O Hub
     to ESI Port (rev 22)
...
00:1a.0 USB controller: Intel Corporation 82801JI (ICH10
    Family) USB UHCI Controller #4
00:1a.1 USB controller: Intel Corporation 82801JI (ICH10
    Family) ...
```

14.6 Listando Todos os Dispositivos de Bloco

Os dispositivos de Entrada e Saída de um computador podem ser divididos em dispositivos de bloco e dispositivos de caracteres. Os dispositivos de caracteres comunicam-se utilizando fluxo de caracteres enquanto os dispositivos de bloco transferem um conjunto de dados durante o processo de comunicação [11]. Teclados e mouses são dispositivos de caracteres e disco rígidos são dispositivos que comunicam-se utilizando blocos de dados.

O comando **lsblk** lista todos os dispositivos de bloco de seu computador. A Listagem 14.8 apresenta o resultado do comando. Neste comando, aparecerá um dispositivo de bloco */dev/sda* e */dev/sdb* e suas respectivas partições.

Listagem 14.8: Listando Dispositivos Bloco

```
decom@tucunare:~$ lsblk
NAME                          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                             8:0    0 816,8G  0 disk
  sda1                          8:1    0   243M  0 part /boot
  sda2                          8:2    0     1K  0 part
  sda5                          8:5    0 816,5G  0 part
    tucunare--vg-root         254:0    0 800,5G  0 lvm  /
    tucunare--vg-swap_1       254:1    0    16G  0 lvm  [SWAP]
sr0                            11:0    1  1024M  0 rom
```

14.7 Verificando Todas as Partições

Partições são organizações lógicas de um disco rígido. Quando instalamos um disco rígido particionamos este disco e instalamos diferentes informações nas mesmas. Com o comando **cat** é possível visualizar as informações sobre as partições em seu disco. A Listagem 14.9 apresenta o resultado do comando.

Listagem 14.9: Imprimindo as Partições

```
[root@musashi ~]$ cat /proc/partitions
major minor  #blocks  name

   8        0  488386584 sda
   8        1     120456 sda1
   8        2   10258432 sda2
   8        3  262023268 sda3
   8        4          1 sda4
   8        5     512000 sda5
   8        6  215469056 sda6
  11        0    1048575 sr0
 253        0    3932160 dm-0
```

```
 253        1   52428800 dm-1
 253        2  159100928 dm-2
   8       16  488386584 sdb
   8       17     204800 sdb1
   8       18  488050672 sdb2
```

Outra maneira de verificar as partições de seu disco é usar o comando fdisk como na Listagem 14.10.

Listagem 14.10: Mostrando as Partições

```
vivas@musashi:~$ sudo fdisk -l
[sudo] password for vivas:
Disk /dev/loop0: 4.7 MiB, 4927488 bytes, 9624 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop1: 88.7 MiB, 92983296 bytes, 181608
   sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop2: 88.4 MiB, 92692480 bytes, 181040
   sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop3: 88.5 MiB, 92778496 bytes, 181208
   sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes




Disk /dev/sda: 836.6 GiB, 898319253504 bytes,
   1754529792 sectors
Units: sectors of 1 * 512 = 512 bytes
```

```
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 3112899F-BFDE-4378-B2A7-65E9B5EE9A83

Device      Start        End    Sectors   Size Type
/dev/sda1    2048       4095       2048     1M BIOS
   boot
/dev/sda2    4096 1754527743 1754523648 836.6G Linux
   filesystem
```

14.8 Listando Dispositivos PCMCIA

O comando **lspcmcia** lista todos os dispositivos PCMCIA em seu computador. A Listagem 14.12 apresenta a sintaxe do comando. Neste caso não há dispositivos PCMCIA no computador. Para utilizar este comando é necessário instalar o pacote pcmciautils como na Listagem 14.11.

Listagem 14.11: Instalando o pacote pacmciautils

```
root@masamune:~# apt-get install pcmciautils
Lendo listas de pacotes... Pronto
...
```

Para listar os dispositivos compatíveis com o PCMCIA basta digitar o comando lspcmcia como na Listagem 14.12.

Listagem 14.12: Listando Dispositivos PCMCIA

```
root@masamune:~# lspcmcia
```

14.9 Obtendo Informações sobre a Memória

O comando **free** lista a quantidade de memória disponível em seu computador. A Listagem 14.13 apresenta o resultado do comando.

Listagem 14.13: Listando Informações sobre a Memória

```
vivas@musashi:~$ free
              total        used        free
                 shared  buff/cache   available
Mem:      132008772     4760316   118222252
   2876     9026204   126175672
Swap:       8388604           0     8388604
```

Total corresponde ao total de memória instalado em seu computador, used corresponde a memória usada, free é a quantidade de memória disponível, shared é a quantidade de memória compartilhada e buff/cache mostra a soma de buffers e cache.

Para verificar estas informações em mega bytes utilize ao opção -m como na Listagem 14.14.

Listagem 14.14: Listando Informações sobre a Memória

```
$ free -m
              total        used        free
                 shared  buff/cache   available
Mem:         128914        1100       96908
   2       30906      126767
Swap:          8191           0        8191
```

Outra opção é verificar o arquivo meminfo no diretório /proc como na Listagem 14.15.

Listagem 14.15: Listando Informações sobre a Memória

```
$$ cat /proc/meminfo
MemTotal:        132008792 kB
MemFree:         99238216 kB
MemAvailable:    129814456 kB
Buffers:            98964 kB
Cached:          30232540 kB
SwapCached:             0 kB
Active:           3130480 kB
Inactive:        27704624 kB
Active(anon):      505916 kB
Inactive(anon):       384 kB
Active(file):     2624564 kB
Inactive(file): 27704240 kB
Unevictable:            0 kB
Mlocked:                0 kB
SwapTotal:        8388604 kB
SwapFree:         8388604 kB
Dirty:                152 kB
Writeback:              0 kB
AnonPages:         503720 kB
Mapped:            134472 kB
Shmem:               2672 kB
Slab:             1599376 kB
SReclaimable:     1316564 kB
SUnreclaim:        282812 kB
```

```
KernelStack:        18768 kB
PageTables:         13288 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:     74393000 kB
Committed_AS:    14630804 kB
VmallocTotal:    34359738367 kB
VmallocUsed:            0 kB
VmallocChunk:           0 kB
HardwareCorrupted:      0 kB
AnonHugePages:          0 kB
ShmemHugePages:         0 kB
ShmemPmdMapped:         0 kB
CmaTotal:               0 kB
CmaFree:                0 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
DirectMap4k:       531680 kB
DirectMap2M:     133672960 kB
```

O comando **vmstat** é outra opção para buscar informações sobre a memória. A Listagem 14.16 apresenta o resultado do comando.

Listagem 14.16: Listando Informações sobre a Memória

```
$ vmstat -s
    132008792 K total memory
      1120248 K used memory
      3130632 K active memory
     27704600 K inactive memory
     99240424 K free memory
        98996 K buffer memory
     31549124 K swap cache
      8388604 K total swap
            0 K used swap
      8388604 K free swap
       104122 non-nice user cpu ticks
         5852 nice user cpu ticks
       105865 system cpu ticks
   1456054233 idle cpu ticks
       183114 IO-wait cpu ticks
```

```
            0 IRQ cpu ticks
         1403 softirq cpu ticks
            0 stolen cpu ticks
      1543822 pages paged in
     40500836 pages paged out
            0 pages swapped in
            0 pages swapped out
     54252082 interrupts
     40337693 CPU context switches
   1567423902 boot time
        89702 forks
```

Para listar os ranges de memória disponível utilize o comando **lsmem** como na Listagem 14.17.

Listagem 14.17: Listando Informações sobre a Memória

```
$ lsmem
RANGE                                  SIZE  STATE
   REMOVABLE BLOCK
0x0000000000000000-0x000000007fffffff    2G online
          no     0
0x0000000100000000-0x00000007ffffffff   28G online
         yes  2-15
0x0000000800000000-0x000000087fffffff    2G online
          no    16
0x0000000880000000-0x0000000c7fffffff   16G online
         yes 17-24
0x0000000c80000000-0x000000107fffffff   16G online
          no 25-32
0x0000001080000000-0x000000157fffffff   20G online
         yes 33-42
0x0000001580000000-0x000000187fffffff   12G online
          no 43-48
0x0000001880000000-0x0000001effffffff   26G online
         yes 49-61
0x0000001f00000000-0x000000207fffffff    6G online
          no 62-64

Memory block size:         2G
Total online memory:     128G
Total offline memory:      0B
```

Para obter informações sobre a memória RAM de seu computador utilize o comando **dmidecode** como na Listagem 14.18. Este comando necessita de acesso privilegiado.

Listagem 14.18: Listando Informações sobre a Memória

```
root@debian:~# dmidecode -t 17
# dmidecode 3.0
Getting SMBIOS data from sysfs.
SMBIOS 2.6 present.

Handle 0x1100, DMI type 17, 28 bytes
Memory Device
        Array Handle: 0x1000
        Error Information Handle: Not Provided
        Total Width: 64 bits
        Data Width: 64 bits
        Size: 2048 MB
        Form Factor: DIMM
        Set: None
        Locator: DIMM 1
        Bank Locator: Not Specified
        Type: DDR3
        Type Detail: Unbuffered (Unregistered)
        Speed: 1333 MHz
        Manufacturer: 019400000194
        Serial Number: 00A8044F
        Asset Tag: 06110300
        Part Number: SH564568FH8N6PHSFG
        Rank: 1

Handle 0x1101, DMI type 17, 28 bytes
Memory Device
        Array Handle: 0x1000
        Error Information Handle: Not Provided
        Total Width: 64 bits
        Data Width: 64 bits
        Size: 2048 MB
        Form Factor: DIMM
        Set: None
        Locator: DIMM 2
        Bank Locator: Not Specified
        Type: DDR3
        Type Detail: Unbuffered (Unregistered)
        Speed: 1333 MHz
        Manufacturer: 019400000194
        Serial Number: 00A8044E
        Asset Tag: 06110300
```

```
        Part Number: SH564568FH8N6PHSFG
        Rank: 1

Handle 0x1102, DMI type 17, 28 bytes
Memory Device
        Array Handle: 0x1000
        Error Information Handle: Not Provided
        Total Width: 64 bits
        Data Width: 64 bits
        Size: 2048 MB
        Form Factor: DIMM
        Set: None
        Locator: DIMM 3
        Bank Locator: Not Specified
        Type: DDR3
        Type Detail: Unbuffered (Unregistered)
        Speed: 1333 MHz
        Manufacturer: 019400000194
        Serial Number: 00A80459
        Asset Tag: 06110300
        Part Number: SH564568FH8N6PHSFG
        Rank: 1

Handle 0x1103, DMI type 17, 28 bytes
Memory Device
        Array Handle: 0x1000
        Error Information Handle: Not Provided
        Total Width: 64 bits
        Data Width: 64 bits
        Size: 2048 MB
        Form Factor: DIMM
        Set: None
        Locator: DIMM 4
        Bank Locator: Not Specified
        Type: DDR3
        Type Detail: Unbuffered (Unregistered)
        Speed: 1333 MHz
        Manufacturer: 019400000194
        Serial Number: 00A80451
        Asset Tag: 06110300
        Part Number: SH564568FH8N6PHSFG
        Rank: 1
```

Outros comandos podem ser utilizados para verificação de informações

sobre memória: top e htop. O comando htop necessita de ser instalado e o top é nativo do sistema operacional. As Figuras 14.1 e 14.2 apresentam as telas deste comando.

Figura 14.1: Comando top

Figura 14.2: Comando htop

14.10 Listando Todos os Dispositivos de Hardware

O comando **hwinfo** é utilizado para listar informações sobre todos os dispositivos de Hardware de seu computador. O **hwinfo** não vem instalado nas distribuições de Linux. O processo de instalação pode ser visualizado na Listagem 14.19. Outra opção é o uso do comando **lshw**.

Listagem 14.19: Instalando hwinfo

```
vivas@musashi:~$ sudo apt-get install hwinfo
[sudo] password for vivas:
Reading package lists... Done
Building dependency tree
Reading state information... Done
...
```

Os comandos **lshw** e **hwinfo** realizam o mesmo procedimento. A Listagem 14.20 apresenta o resultado do comando. Devido o tamanho da listagem optamos por omitir alguns resultados. Para uma versão completa do comando rode o mesmo como super usuário.

Listagem 14.20: Analisando os Dispositivos de Hardware

```
vivas@musashi:~$ sudo hwinfo
```

Se quiser uma versão resumida do relatório utilize a Listagem 14.21.

Listagem 14.21: Versão Resumida do Relatório do lshw

```
vivas@musashi:~$ sudo hwinfo --short
cpu:
                          Intel(R) Xeon(R) CPU           E7530  @ 1.87GHz, 1995 MHz
                          Intel(R) Xeon(R) CPU           E7530  @           16550A

...
```

14.11 Comando eject

O comando **eject** pode ser utilizado para ejetar CDROMs ou DVDs de seu computador. A sintaxe do comando pode ser visualizada na Listagem 14.22.

Listagem 14.22: Comando para remover CDROM

```
vivas@musashi:~$ eject
...
```

Capítulo 15

Comandos de Data e Hora

Sumário

15.1 Comando Date

O comando **date** exibe a hora e data do sistema. A sintaxe do comando é igual para o Linux e o Mac. A Listagem 15.1 apresenta o resultado do comando **date**.

Listagem 15.1: Visualizando Data e hora

```
$ date
qui set  5 17:11:46 -03 2019
```

15.2 Calendário do Mês Corrente

O comando cal exibe um calendário em formato texto no terminal. Pode-se definir o ano e nesse caso todos os meses serão apresentados. Pode-se definir o mês específico. Se nenhum parâmetro for passado, o mês atual é exibido. A Listagem 15.2 apresenta o comando **cal** .

Listagem 15.2: Comando cal

```
$ cal
   Setembro 2019
do se te qu qu se sá
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
```

15.3 Calendário de um Mês Específico

Você pode imprimir um mês específico como na Listagem 15.3

Listagem 15.3: Comando cal

```
$ cal -m Janeiro
    Janeiro 2019
do se te qu qu se sá
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

15.4 Calendário de um Ano Específico

Para imprimir o calendário de um ano específico utilize a opção -y como na Listagem 15.4.

Listagem 15.4: Calendário de um Ano Específico

```
$ cal -y 1973
                              1973
       Janeiro               Fevereiro               Março
do se te qu qu se sá  do se te qu qu se sá  do se te qu qu se sá
    1  2  3  4  5  6               1  2  3               1  2  3
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   4  5  6  7  8  9 10
14 15 16 17 18 19 20  11 12 13 14 15 16 17  11 12 13 14 15 16 17
21 22 23 24 25 26 27  18 19 20 21 22 23 24  18 19 20 21 22 23 24
28 29 30 31           25 26 27 28           25 26 27 28 29 30 31

        Abril                 Maio                  Junho
do se te qu qu se sá  do se te qu qu se sá  do se te qu qu se sá
 1  2  3  4  5  6  7         1  2  3  4  5                  1  2
 8  9 10 11 12 13 14   6  7  8  9 10 11 12   3  4  5  6  7  8  9
15 16 17 18 19 20 21  13 14 15 16 17 18 19  10 11 12 13 14 15 16
22 23 24 25 26 27 28  20 21 22 23 24 25 26  17 18 19 20 21 22 23
29 30                 27 28 29 30 31        24 25 26 27 28 29 30

        Julho                Agosto               Setembro
do se te qu qu se sá  do se te qu qu se sá  do se te qu qu se sá
 1  2  3  4  5  6  7            1  2  3  4                     1
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   2  3  4  5  6  7  8
15 16 17 18 19 20 21  12 13 14 15 16 17 18   9 10 11 12 13 14 15
22 23 24 25 26 27 28  19 20 21 22 23 24 25  16 17 18 19 20 21 22
29 30 31              26 27 28 29 30 31     23 24 25 26 27 28 29
                                            30

       Outubro               Novembro              Dezembro
do se te qu qu se sá  do se te qu qu se sá  do se te qu qu se sá
    1  2  3  4  5  6               1  2  3                     1
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   2  3  4  5  6  7  8
14 15 16 17 18 19 20  11 12 13 14 15 16 17   9 10 11 12 13 14 15
21 22 23 24 25 26 27  18 19 20 21 22 23 24  16 17 18 19 20 21 22
28 29 30 31           25 26 27 28 29 30     23 24 25 26 27 28 29
```

15.5 Comando calendar

O comando **calendar** imprime os eventos que ocorreram a partir da data corrente (Listagem 15.5).

Listagem 15.5: Comando calendar

```
$ calendar
set 05  King Louis XIV of France born, 1638
set 05  Raquel Welch born, 1942
set 05  US President Kennedy orders resumption of
   underground nuclear tests, 1961
set 05  The first Continental Congress was convened in
    Philadelphia, 1774
set 05  Backports provided as an official Debian
   service, 2010
set 05  Mark Robert Vaughan Murray <markm@FreeBSD.org>
    born in Harare, Mashonaland, Zimbabwe, 1961
```

```
set 05  Adrian Harold Chadd <adrian@FreeBSD.org> born
   in Perth, Western Australia, Australia, 1979
set 05  Rodrigo Osorio <rodrigo@FreeBSD.org> born in
   Montevideo, Uruguay, 1975
set 05  N'oubliez pas les Raïssa !
set 05  Septembre humide,
        Pas de tonneau vide.
set 05  Entführung und Ermordung von Arbeitgeberprä
   sident Schleyer,
        Entführung einer Lufthansa-Maschine nach
           Mogadischu, 1977
set 05  Viktor, Lõrinc
set 06  Word is received that Perry has reached the
   North Pole and died, 1909
set 06  149 Pilgrims set forth from England aboard the
    Mayflower, 1620
set 06  First Star Trek episode (The Man Trap) aired
   1966
set 06  US President McKinley shot, 1901
set 06  Somhlolo in Swaziland
set 06  Defense of Pakistan Day in Pakistan
set 06  Unification of Bulgaria
set 06  Nace el investigador argentino Luis Federico
   Leloir, 1906
set 06  Prins Claus (1925 - 2002)
set 06  Eric Joyner <erj@FreeBSD.org> born in Fairfax,
    Virginia, United States, 1991
set 06  Bonne fête aux Bertrand !
set 06  Zakariás
```

15.6 Comando time

O comando **time** pode ser utilizado para calcular o tempo de execução de programas. A Listagem 15.6 apresenta um exemplo do comando time [4].

Listagem 15.6: Comando time

```
$ time wget https://cdn.kernel.org/pub/linux/kernel/v4
   .x/linux-4.19.9.tar.xz
--2019-09-05 17:31:10--  https://cdn.kernel.org/pub/
   linux/kernel/v4.x/linux-4.19.9.tar.xz
Resolvendo cdn.kernel.org (cdn.kernel.org)...
   151.101.5.176, 2a04:4e42:16::432
```

```
Conectando-se a cdn.kernel.org (cdn.kernel.org)
   |151.101.5.176|:443... conectado.
A requisição HTTP foi enviada, aguardando resposta...
   200 OK
Tamanho: 103135232 (98M) [application/x-xz]

linux 4.19.9.tar.xz 100% 98,36M  3,42MB/s    em 28s

real    0m33,334s
user    0m3,344s
sys     0m1,664s
```

O resultado é apresentado o tempo nos modos real, user e sys. O real corresponde o tempo total entre o início e o final do comando. O parâmetro user mede a quantidade de tempo gasto no modo usuário e sys o tempo gasto no modo kernel.

O comando **time** é utilizado para mostrar o tempo de execução de um script ou processo. Este comando calcula o tempo utilizado pelo programa no modo usuário, no modo kernel e a quantidade de tempo realmente utilizado. A quantidade de tempo total utilizado é sempre maior do que a soma do tempo no modo Kernel mais o tempo no modo usuário. Ele envolve todos os tempos: processamento de interrupção e tempo de espera na fila de processos prontos ou aguardando o processamento de entrada e saída. A Listagem 15.7 calcula o tempo gasto pelo comando date. Neste caso ele gastou 2 ms de tempo no modo kernel, 1 ms no modo usuário e o tempo total de 4 ms.

Listagem 15.7: Calculando Tempo de Execução de um Programa ou Script

```
time date
Sex  4 Set 2015 13:54:02 BRT

real    0m0.004s
user    0m0.001s
sys     0m0.002s
```

Vamos dar o usar o comando da Listagem 15.7 em um notebook rodando Fedora. Os dois tem processadores semelhantes, mas o primeiro roda Mac OSX e o segundo Fedora. O resultado do comando é apresentado na Listagem 15.8. Como pode ser visualizado o notebook rodando Fedora teve um desempenho melhor do que o Mac OSX.

Listagem 15.8: Calculando Tempo de Execução de um Programa ou Script

```
time date
Sex Set  4 14:13:13 BRT 2015
```

```
real    0m0.002s
user    0m0.000s
sys     0m0.002s
```

15.7 Comando timedatectl

O comando **timedatectl** imprime dados sobre hora local, hora universal, tempo RTC, time zone e o tipo de sincronização de hora utilizado (Listagem 15.9).

Listagem 15.9: Comando timedatectl

```
$ timedatectl
               Local time: qui 2019-09-05 17:40:58 -03
           Universal time: qui 2019-09-05 20:40:58 UTC
                 RTC time: qui 2019-09-05 20:40:58
                Time zone: America/Sao_Paulo (-03,
                   -0300)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Capítulo 16

Compactando e Descompactando Arquivos e Diretórios

Sumário

16.1 Comando tar

O comando **tar** pode ser utilizado para compactar arquivos ou diretórios. A Listagem 16.1 apresenta o procedimento para instalação do tar.

Listagem 16.1: Instalando comando tar

```
$ apt install tar
ou
$ sudo apt install tar
```

A Listagem 16.3 apresenta o comando para compactar o diretório de Imagens. As opções têm o seguinte significado:

- c - compactar
- v - imprimir na tela o resultado do comando
- f - indicar o nome do arquivo

Listagem 16.2: Compactando um Diretório

```
$ tar -cvf Imagens.tar Imagens/
Imagens/
Imagens/Captura de tela_2019-09-05_21-00-52.png
Imagens/Captura de tela_2019-09-05_21-00-12.png
```

Para descompactar o arquivo trocamos a opção c por x como na Listagem 16.3.

Listagem 16.3: Descompactando um Arquivo

```
$ tar -xvf Imagens.tar
Imagens/
Imagens/Captura de tela_2019-09-05_21-00-52.png
Imagens/Captura de tela_2019-09-05_21-00-12.png
```

16.2 Compactando com Gzip

Para instalar o gzip basta utilizar o comando da Listagem 16.4.

Listagem 16.4: Instalando comando tar

```
$ apt install gzip
ou
$ sudo apt install gzip
```

Para compactar um arquivo com gzip basta utilizar o comando da Listagem 16.5.

Listagem 16.5: Compactando de Arquivos com gzip

```
$ gzip -c arquivo.txt > arquivo.gz
```

Para descompactar utilizamos o comando da Listagem 16.6.

Listagem 16.6: Descompactando Arquivos com gzip

```
$ gzip -d arquivo.gz
```

Você pode compactar os arquivos de um diretório utilizando a opção -r como na Listagem 16.7.

Listagem 16.7: Compactando Diretórios com gzip

```
$ gzip -r diretorio/
```

Para compactar um diretório podemos utilizar o comando tar com o gzip como na Listagem 16.8.

Listagem 16.8: Compactando Diretórios com tar e gzip

```
$ tar -czvf Compressao.tar.gz Compressao/
Compressao/
Compressao/arquivo.gz
Compressao/arquivo
Compressao/orginal.txt
```

Para descompactar utilize o comando como na Listagem 16.9.

Listagem 16.9: Descompactando Diretórios com tar e gzip

```
$ tar -xzvf Compressao.tar.gz
```

16.3 Compactando com bzip2

Outra opção é utilizar o bzip2 para comprimir arquivos. O processo de instalação está disponível na Listagem 16.10.

Listagem 16.10: Instalando comando bzip2

```
$ apt install bzip2
ou
$ sudo apt install bzip2
```

Para compactar um arquivo com bzip2 basta utilizar o comando da Listagem 16.11.

Listagem 16.11: Compactando de Arquivos com bzip2

```
$ bzip2 arquivo
```

Para descompactar utilizamos os comandos da Listagem 16.12.

Listagem 16.12: Descompactando Arquivos com bzip2

```
$ bzip2 -d arquivo.bz2
ou
$ bunzip2 arquivo.bz2
```

Para compactar um diretório podemos utilizar o comando tar com o bzip2 como na Listagem 16.13.

Listagem 16.13: Compactando Diretórios com tar e bzip2

```
$ tar -cjvf Compressao.tar.bz2 Compressao/
Compressao/
Compressao/arquivo.gz
Compressao/arquivo
Compressao/orginal.txt
```

Para descompactar utilize o comando como na Listagem 16.14.

Listagem 16.14: Descompactando Diretórios com tar e bzip2

```
$ tar -xjvf Compressao.tar.gz
```

16.4 Compactando com xz

Outra opção é utilizar o xz para comprimir arquivos. O processo de instalação está disponível na Listagem 16.15.

Listagem 16.15: Instalando comando xz

```
$ apt install xz-utils
ou
$ sudo apt install xz-utils
```

Para compactar um arquivo com xz basta utilizar o comando da Listagem 16.16.

Listagem 16.16: Compactando de Arquivos com xz

```
$ xz arquivo
```

Para descompactar utilizamos os comandos da Listagem 16.17.

Listagem 16.17: Descompactando Arquivos com xz

```
$ xz -d arquivo.xz
```

Para compactar um diretório podemos utilizar o comando tar com o xz como na Listagem 16.18.

Listagem 16.18: Compactando Diretórios com tar e xz

```
$ tar -cJvf Compressao.tar.xz Compressao/
Compressao/
Compressao/arquivo.gz
Compressao/arquivo
Compressao/orginal.txt
```

Para descompactar utilize o comando como na Listagem 16.19.

Listagem 16.19: Descompactando Diretórios com tar e xz

```
$ tar -xJvf Compressao.tar.xz
```

16.5 Comparando os algoritmos

Para comparar os algoritmos utilizamos o mesmo arquivo e aplicamos os diferentes algoritmos de compressão. A Listagem 16.20 apresenta os resultados obtidos. O arquivo original tinha o tamanho de 11 Mbytes e foram obtidos os seguintes resultados:

- gz - 1,5 Mbytes
- bzip2 - 1,3 Mbytes
- xz - 912Kbytes

Listagem 16.20: Comparando os Algoritmos de Compressão

```
$ du -kh *
1,3M    arquivo.bz2
1,5M    arquivo.gz
912K    arquivo.xz
11M     orginal
```

Capítulo 17

Editor de Texto VIM

Sumário

17.1 Introdução

Existem diversos editores de texto no Linux dentre eles destacam-se o VI , VIM , pico , emacs e o nano .

VIM ou VI improved (melhorado) é um editor robusto e muito versátil presente em todos os sistemas derivados do Unix. Este editor é muito útil quando trabalhamos remotamente ou estamos utilizando o terminal do Linux. A tela inicial do editor VIM é minimalista como apresentado na Figura 17.1.

```
                  VIM - Vi IMproved

                   version 8.0.1453
               by Bram Moolenaar et al.
  Modified by pkg-vim-maintainers@lists.alioth.debian.org
         Vim is open source and freely distributable

               Help poor children in Uganda!
      type  :help iccf<Enter>       for information

      type  :q<Enter>               to exit
      type  :help<Enter>  or  <F1>  for on-line help
      type  :help version8<Enter>   for version info
```

Figura 17.1: Tela Inicial do VIM

17.2 Instalação

O procedimento de instalação do editor VIM é bem simples. Para instalar utilize o código da Listagem 17.1.

Listagem 17.1: Instalando VIM

```
$ sudo apt install vim
```

Para instalar em outros Sistemas Operacionais faça o download (https://www.vim.org/) e instale de acordo com as instruções disponíveis no site.

17.3 Criando Arquivo com vim

O primeiro passo para utilizar o VIM é abrir o terminal e digitar o comando vim e nome do arquivo como na Listagem 17.2.

Listagem 17.2: Criando arquivo com VIM

```
$ vim arquivo1.txt
```

O resultado será a tela da Figura 17.2.

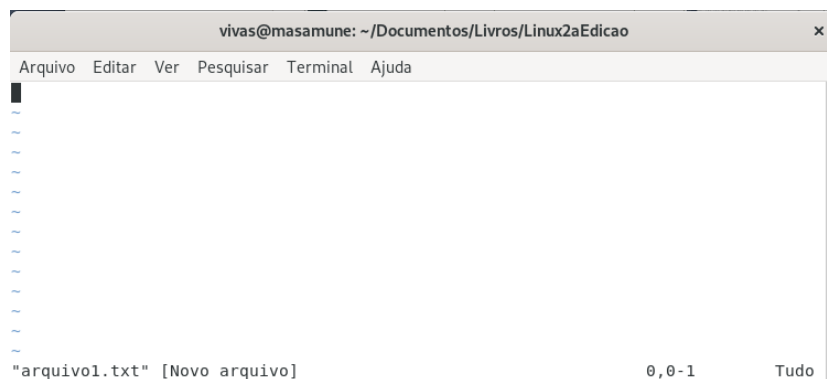



Figura 17.2: Tela do VIM

Para sair do editor digite **:q** .

17.4 Modos de Interação com o Usuário

Nesse editor existem 3 modos de interação com o usuário: modo de comando, de visualização e modo de edição.

Quando inicializamos o editor, ele é iniciado no modo de comandos. Para podermos modificar o texto, precisamos entrar no modo de edição (inserção). Para realizar esta operação digite a tecla **i** . Você observará que surge a palavra **INSERT** ou **INSERÇÃO** no canto inferior esquerdo da tela (Figura 17.3).

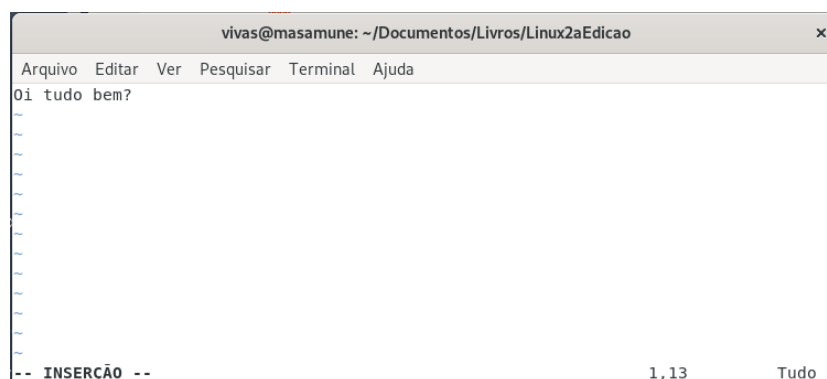



Figura 17.3: Modo de Inserção

O outro modo mais usado no editor é o de comando. Conseguimos ir

para o mesmo pressionando a tecla **ESC** . Se você não tem certeza se está no modo de comando, saberá que está no referido modo ao pressionar tal tecla e ouvir um bip agudo, indicando que o VIM já se encontra no modo de comando (Figura 17.4).

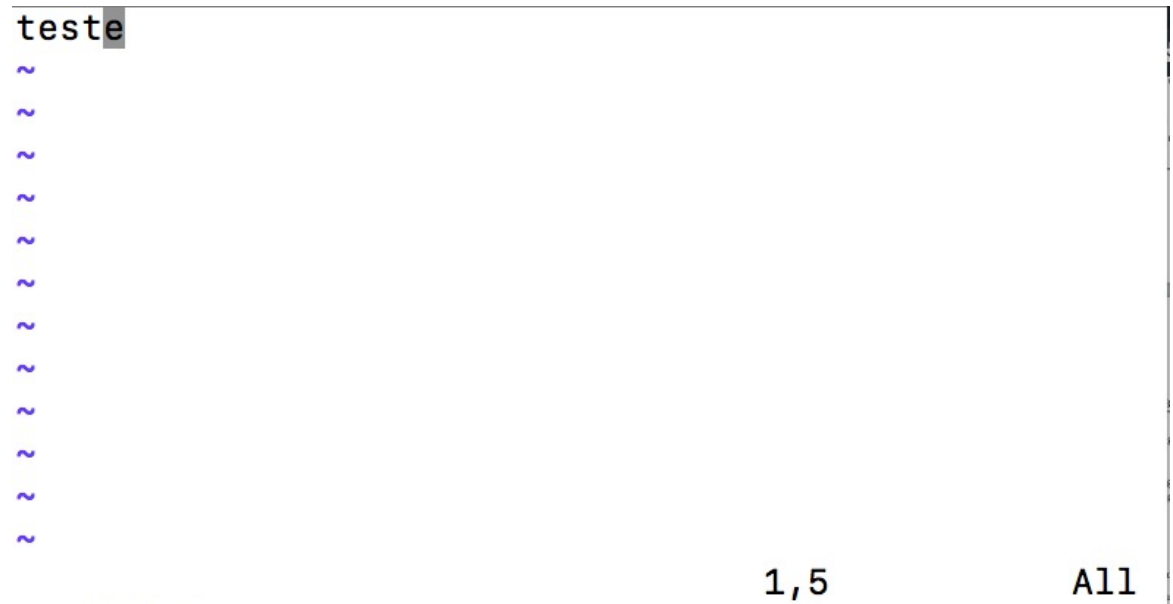



Figura 17.4: Retornando ao Modo de Comandos

Existem diversos tipos de comandos, sendo eles divididos em comandos explícitos e implícitos. Os comandos explícitos são passados ao editor pelo usuário e visualizados no canto inferior esquerdo da tela à medida que são inseridos. A definição de um comando explícito é iniciada sempre com dois pontos(: - modo de comando estendido). A seguir, iremos mencionar os principais comandos desse tipo.

17.5 Criando um Arquivo com Editor VIM

O usuário tem duas opções para iniciar o editor VIM: (a) passando o nome de um arquivo ou (b) não utilizando nenhum argumento. Se o arquivo existir em seu diretório o editor irá mostrar o conteúdo do arquivo e caso não exista o arquivo o editor irá criar um novo arquivo. A Listagem 17.3 apresenta a sintaxe para abrir um arquivo utilizando o VIM. Substitua <**nome do arquivo**> pelo nome do arquivo que você deseja abrir.

Listagem 17.3: Abrindo um arquivo com VIM

```
$ vim   <nome do arquivo>
```

Vamos supor que seu arquivo tenha o nome de cidades.txt e esteja no diretório corrente. Para abrir o arquivo digite o comando da Listagem 17.4.

Listagem 17.4: Abrindo um arquivo com VIM

```
$ vim  cidades.txt
```

Como resultado será mostrado o conteúdo do arquivo que estava no seu diretório como na Listagem 17.5. No final do arquivo irá aparecer o nome do arquivo, o número de linhas e o número de caracteres do arquivo.

Listagem 17.5: Abrindo um arquivo com VIM

```
Diamantina
Lavras
Gouveia



~
~
"cidades.txt" 4L, 28C
```

Para fechar o arquivo basta digitar o comando **:q** , como previamente explicado.

17.6 Inserindo Texto

Para inserir texto no VIM é necessário entrar no modo de edição. Como não existem menus para acessar via mouse é necessário digitar o comando que indica ao VIM que o texto será digitado. Vamos falar dois modos do editor a) modo de comandos b) modo de edição. O modo de comandos é utilizado para você digitar comandos como salvar, sair, fazer buscas, etc.. O modo de edição é responsável pela inserção de texto no editor e é acessado pelo comando **i** .

Listagem 17.6: Inserir Texto Parte 1

```
andarilho-2:~ alessandrovivas$ vim

~
~
~
~                              VIM - Vi IMproved
~
~                              version 7.4.8056
~                          by Bram Moolenaar et al.
```

```
~                    Vim is open source and freely
   distributable
~
~                              Help poor children in Uganda!
~                  type  :help iccf<Enter>        for
   information
~
~                  type  :q<Enter>                to exit
~                  type  :help<Enter>  or  <F1>   for on-
   line help
~                  type  :help version7<Enter>    for
   version info
~
```

Para digitar o texto vamos então digitar o comando para entrar no modo de edição (**i**) como na Listagem 17.7. Quando entra no modo de inserção aparece no final do editor a mensagem **INSERT**

Listagem 17.7: Inserir Texto Parte 2

```
~
~
~
~
~
~                                     VIM - Vi IMproved
~
~                                     version 7.4.8056
~                               by Bram Moolenaar et al.
~                    Vim is open source and freely
   distributable
~
~                               Sponsor Vim development!
~                  type  :help sponsor<Enter>     for
   information
~
~                  type  :q<Enter>                to exit
~                  type  :help<Enter>  or  <F1>   for on-
   line help
~                  type  :help version7<Enter>    for
   version info
~
~
~
```

```
~
~
-- INSERT --
```

Agora é só digitar o texto como foi feito na Listagem 17.8 .

Listagem 17.8: Inserir Texto Parte 3

```
Ubuntu
Suse
Debian

~
~
-- INSERT --
```

Agora preciso voltar para o modo de comandos para salvar o arquivo e dar o nome ao mesmo. Para voltar para o modo de comandos é necessário digitar o comando **ESC** como na Listagem 17.9. Perceba que não aparece mais a mensagem **INSERT** no final do arquivo.

Listagem 17.9: Inserir Texto Parte 4

```
Ubuntu
Suse
Debian

~
~
~
~
```

Agora é possível salvar o arquivo com o nome de por exemplo distribuicoes.txt através do comando **:w** como na Listagem 17.10.

Listagem 17.10: Inserir Texto Parte 5

```
Ubuntu
Suse
Debian

~
:w distribuicoes.txt
```

Após digitar o comando **:w** aparecerá uma informação do número de linhas, caracteres e o nome de arquivo escolhido como na Listagem 17.11.

Listagem 17.11: Inserir Texto Parte 6

```
Ubuntu
Suse
Debian

~
~
"distribuicoes.txt" [New] 4L, 20C written
```

Para sair do editor basta utilizar o comando **:q** como na Listagem 17.12.

Listagem 17.12: Inserir Texto Parte 7

```
Ubuntu
Suse
Debian

~
:q
```

17.7 Salvando um arquivo

Para salvar o conteúdo editado num arquivo com nome <nome do arquivo>, você deve usar o comando w (write) :

Listagem 17.13: Salvando o arquivo

```
:w <nome do arquivo>
```

Se o nome do arquivo já se encontra definido, ou seja, você abriu um arquivo existente ou chamou o editor passando para o mesmo o nome do novo arquivo, basta usar o comando **:w** sem nenhum argumento. A definição de um outro nome implica na criação de um novo arquivo com o conteúdo atual. Mas, atenção: é fundamental que você tenha permissão de escrita no diretório em que se encontra. Caso contrário, defina um outro local para armazenar o arquivo detalhando um caminho inteiro/parcial de diretórios, como nos exemplos a seguir:

Listagem 17.14: Salvando um arquivo

```
:w /home/textos/<nome do arquivo>
```

Listagem 17.15: Salvando um arquivo

```
:w ../exercicios/aula1/<nome do arquivo>
```

17.7.1 Saindo do editor

Se entramos no editor, pretendemos alguma hora sair dele. Para isso, existe o comando estendido :q (quit) .

Listagem 17.16: Saindo do Arquivo

```
:q
```

Esse comando só será aceito se as modificações no arquivo já estiverem salvas ou se elas não existiram em momento algum. Caso você queira salvar as alterações feitas no arquivo e ao mesmo tempo sair do editor, é permitida a combinação de comandos. Nessa situação, você deve digitar **:wq** (ou de forma alternativa ZZ, sem os dois pontos).

Aqui o texto será salvo e, logo em seguida, você estará abandonando o VIM. Outra forma equivalente à combinação anteriormente citada é o comando **:x** , que produz o mesmo efeito. Mais uma vez, se não lhe é permitida a gravação do arquivo o comando não terá sucesso. Você pode usar o recurso de imposição ao programa através do uso do caractere exclamação (!). Quando ele é usado, significa que o usuário exige que a ação seja realizada. Dessa maneira, a não permissão de gravação pode ser desconsiderada ao se usar o comando :x!.

Essa sintaxe só irá funcionar caso a proibição seja 'contornável'. Por exemplo, o dono de um arquivo pode retirar sua permissão de gravação via shell. Mesmo assim, ele tem a possibilidade de renomear, apagar, mover ou alterar o conteúdo desse arquivo, utilizando a 'força bruta'. Nesse caso, o usuário (dono do arquivo) poderá executar o comando de gravação forçada por meio do caractere !. A forma mais comum de uso da exclamação é em conjunto com o comando :q. Aqui estamos impondo ao editor uma saída forçada, sem salvar (descartando) modificações feitas no arquivo editado.

Listagem 17.17: Saindo do Arquivo

```
:q!
```

17.8 Alternando entre Arquivos

Assim como um editor convencional, o VIM lhe permite a alternância entre arquivos. Para isso, existe o comando:

Listagem 17.18: Alternando arquivos

```
:e <nome do arquivo>
```

Ele lhe possibilita a edição de um outro arquivo sem a necessidade de se sair do VI (pode-se usar também o comando :e! para forçar a mudança, caso não haja salvo o arquivo atual).

17.8.1 Adicionando outro arquivo

Você pode também querer adicionar no seu arquivo corrente o conteúdo de um outro arquivo, realizando uma operação do tipo append. Basta posicionar o cursor no local onde o novo conteúdo deverá ser inserido e digitar comando **:r** seguido do nome do arquivo como na Listagem 17.19

Listagem 17.19: Adicionando outro arquivo

```
:r <nome do arquivo>
```

17.8.2 Executando comandos no shell

Você pode executar programas e o resultado ser inserido em um arquivo VIM. Para isto, abra um arquivo e depois utilize o comando **!** como na Listagem 17.20.

Listagem 17.20: Executando comandos no shell

```
:! comando de Shell
```

A Listagem 17.21 apresenta o resultado do comando dmesg.

Listagem 17.21: Executando comandos no shell

```
:! dmesg
```

Se quiser inserir o resultado do comando no arquivo utilize o comando **:r!** como na Listagem 17.22.

Listagem 17.22: Executando comandos no shell

```
:r! comando de shell
```

17.9 Copiando Linhas

Para copiar uma linha, posicione o cursor na linha desejada e digite **yy** (no modo de comandos) . Posicione o cursor no local desejado e digite **p** para colar o resultado.

Para copiar múltiplas linhas, posicione o cursor na linha inicial e digite o número de linhas desejadas seguida de **y**. Se deseja copiar 3 linhas, digite **3yy** e depois posicione o cursor no local onde o conteúdo deverá ser copiado e digite **p**.

17.10 Apagando Linhas

Para apagar uma linha posicione o cursor no local desejado e digite **dd** no modo de comandos. Caso deseje apagar mais linhas posicione o cursor na linha inicial e digite o número de linhas seguido de **dd**. Se desejar copiar 3 linhas digite **3dd**.

17.11 Desfazendo Operações

Se você fez alguma operação e deseja voltar ao estado anterior digite **u** .

Capítulo 18

Scripts no Linux

Sumário

18.1 Introdução

Através dos comandos descritos neste livro podemos criar scripts para automatizar diversas tarefas cotidianas. Podemos executar um script diretamente no terminal, ou podemos criar um arquivo contendo o script. Para executar o script demovemos chamar o interpretador e passar o arquivo de script como parâmetro. Veja e o exemplo na Listagem 18.1, onde salvamos o conteúdo do script em um arquivo chamado 'test.sh'.

Listagem 18.1: Script para listar arquivos png

```
$ cat test.sh
ls *.png
$ /bin/bash test.sh
 001.png
 002.png
 ...
```

O mais usual é identificarmos na primeira linha do script quem será o interpretador. Para tanto utiliza-se o sha-bang (#!) na primeira linha do script, seguida do interpretador. Assim poderemos executar o script diretamente chamando-o pelo seu nome. Mas, antes, devemos dar permissão de execução ao arquivo. Veja o exemplo na Listagem 18.2.

Listagem 18.2: Script para listar arquivos png

```
$ cat test.sh
#! /bin/bash
ls *.png
$ ls -l test.sh
-rw-r--r-- 1 leoca leoca 22 out  7 17:48 test.sh
$ chmod u+x test.sh
$ ls -l test.sh
$ ls -l test.sh
-rwxr--r-- 1 leoca leoca 22 out  7 17:48 test.sh
$ ./test.sh
 001.png
 002.png
 ...
```

Verficamos no exemplo que foi atribuído o flag **-x** ao arquivo, permitindo assim a execução pelo usuário proprietário do arquivo (pois utilizamos o parâmetro **-u** no comando **chmod)**. Note também que devemos executar o script usando ./ antes do nome do arquivo, pois o script encontra-se no diretório corrente. Se o diretório que contém o script estiver no **PATH**, bastará utilizar o nome do arquivo do script.

18.2 Variáveis

Um dos elementos básicos de qualquer linguagem de programação são as variáveis. Variáveis são utilizadas para armazenar dados. Veremos nesta secção algumas formas de utilizar variáveis.

Devemos distinguir o nome da variável e o valor a ela associado. No exemplo da Listagem 18.3 definimos uma variável chamada 'testvar' e a ela atribuímos o valor '123'. Para obtermos o valor da variável devemos utilizar o **$**.

Listagem 18.3: Variáveis

```
$ testvar=123
$ echo $testvar
123
```

Ao atribuir ou alterar o valor de uma variável, utilizamos o seu nome, sem o **$**.

Utilizar aspas duplas não interfere na substituição do valor de uma variável, porém utilizar aspas simples não. Veja o exemplo na Listagem 18.4.

Listagem 18.4: Variáveis

```
$ a=123
$ b=$a; echo $b
123
$ b="$a"; echo $b
123
$ b='$a'; echo $b
$a
$ b=${a}; echo ${b}
123
```

Note no exemplo anterior que também utilizamo o nome da variável entre chaves. Esta é a forma completa, ou seja, utilizar sem as chaves é uma forma simplificada.

O exemplo da Listagem 18.5 apresenta atribuição de valores a uma variável. Note que o primeiro resultado pode não ser o que foi pretendido.

Listagem 18.5: Variáveis

```
$ a=120+3; echo $a
120+3
$ let a=120+3; echo $a
123
```

As variáveis no Bash não são segregadas por tipo. De forma geral, elas são strings de caracteres, mas dependendo do contexto é possível realizar operações matemática ou comparações.

No exemplo da Listagem 18.6, iremos atribuir a uma variável o resultado do comando **ls**. Em seguida vamos utilizar duas formas de ver o valor da variável: a primeira sem utilizar aspas dupla e a segunda utilizando. Observe que apenas utilizando aspas duplas teremos as quebras de linha preservadas.

Listagem 18.6: Variáveis

```
$ a=‘ls -l‘
$ echo $a
total 0 -rw-r--r-- 1 leoca leoca 0 out 8 10:28 001.txt
    -rw-r--r-- 1 leoca leoca 0 out 8 10:28 002.txt
$ echo "$a"
total 0
-rw-r--r-- 1 leoca leoca 0 out  8 10:28 001.txt
-rw-r--r-- 1 leoca leoca 0 out  8 10:28 002.txt
```

18.3 Algumas variáveis internas

Vamos mostrar nesta seção apenas algumas variáveis internas que podem ser úteis na elaboração de um script. A Tabela 18.1 descreve algumas variáveis.

Tabela 18.1: Variáveis internas

variável	significado
$HOME	diretório ‘home’ do usuário
$LC_COLLATE	conjunto de parâmetros que definem a língua, região e formatação
$PWD	diretório de trabalho
$OLDPWD	diretório de trabalho anterior
$PATH	lista de diretórios no caminho para execução
$PPID	ID do processo
$SECONDS	tempo de execução (em segundos) do script
$UID	ID do usuário
$0, $1, $2, etc.	parâmetros posicionais
$#	número de argumentos
"$*"ou "$@"	todos os parâmetros
$!	ID do último processo
$_	argumento do último comando
$?	status de saída
$$	ID do próprio script

18.4 Parâmetros em um script

Ao definir um script, podemos passar parâmetros posicionais. Os argumentos posicionais serão referenciados pelo número inteiro indicando a posição. **$0** indicará a string utilizada para invocar o script (nome do script possivelmente precedido por ./), **$1** será a primeira string após o nome do script, **$2** a segunda string, e assim por diante. Veja o exemplo simples na Listagem 18.7.

Listagem 18.7: Argumentos de um script

```
$ cat tst.sh
#! /bin/bash
echo "nome_do_script=$0"
echo "numero_de_argumentos=$#"
echo "arg1=$1"
echo "arg2=$2"
$ chmod +x tst.sh
$ ./tst.sh foo bar
nome_do_script=./tst.sh
numero_de_argumentos=2
arg1=foo
arg2=bar
```

Algumas vezes queremos passar um número indeterminado de argumentos para o script. Neste caso, podermos utilizar o comando **shift** para passar ao próximo argumento. Para ilustrar sua utilização, iremos utilizar um teste condicional. Um teste condicional é feito com a expressão que se deseja testar entre colchetes. Iremos realizar um loop utilizando **until** e testar se o parâmetro não é vazio, ou seja, se a string não possui comprimento nulo (zero). Para tanto, utilizaremos o **-z**. Veja o exemplo na Listagem 18.8.

Listagem 18.8: Script com um número indefinido de argumentos

```
$ cat tst2.sh
#! /bin/bash
echo -n "BEGIN"
until [ -z "$1" ]
do
  echo -n "$1"
  shift
done
echo -n "END"
$ ./tst2.sh foo bar bar2 foo2 end
BEGINfoobarbar2foo2endEND
```

18.5 Finalização

Nos exemplos vistos anteriormente, não definimos explicitamente o fim dos scripts, mas ao achegar no final eles são finalizados, como observamos ao executá-los. Porém, podemos definir explicitamente o fim de um script utilizando o comando **exit**

O fim de execução de um script retorna um status. Quando há sucesso, retorna-se 0 (zero), e quando há erro, retorna-se algo diferente de zero, sendo que o valor de retorno é usualmente utilizado para indicar qual o erro.

Nos casos em que não há explicitamente a chamada pelo comando **exit**, o retorno ao final do script será do valor de retorno do último comando executado pelo script. Se chamarmos o comando **exit** sem explicitar o valor de retorno, teremos o mesmo comportamento.

Após finalizar um script, o valor de retorno da saída será armazenada na variável **$?**.

Vamos executar o mesmo script da Listagem 18.8 e em seguida verificar o status de saída. Veja o resultado na Listagem 18.9.

Listagem 18.9: Status de saída de um script

```
$ ./tst2.sh foo bar bar2 foo2 end && echo $?
BEGINfoobarbar2foo2endEND0
```

Alguns valores de retorno de saída possuem significado especial. A tabela 18.2 apresentas esses valores e seu significado.

Listagem 18.10: Status de saída de um script

```
$ cat tst3.sh
#! /bin/bash
echo -n "BEGIN"
sleep 2m
echo -n "END"
$ ./tst3.sh foo bar bar2 foo2 end
BEGIN^C
$ echo $?
130
```

Na Listagem 18.11 veremos o resultado de retorno de saída quando um processo é forçado a terminar utilizando o **kill -9**.

Listagem 18.11: Status de saída de um script

```
$ cat tst3.sh
#! /bin/bash
echo "PID: $$"
echo "PPID: $PPID"
```

Tabela 18.2: Códigos de saída

código	significado UNIX
0	sucesso
1	erro genérico
2	error de sintaxe
126	comando invocado não pode ser executado
127	comando não encontrado
128	argumento inválido para sair
128+n	erro fatal n
130	script foi finalizado com Ctrl+C

```
echo "UID: $UID"
sleep 10m
echo "END"
$ ./tst3.sh
PID: 15726
PPID: 18687
UID: 1000
# o script entra em modo sleep...
# enquanto isso vamos em outro terminal
# onde executamos os seguintes comandos:
$ SCRIPTPID=$(ps -ef | grep 'tst3.sh' | grep -v 'grep'
    | tr -s ' ' | cut -d ' ' -f2)
$ kill -9 $SCRIPTPID
# ao retornar para o terminal original veremos
# que o script foi interrompido e podemos então
# verificar o código de saída (128+9=137)
Killed
$ echo $?
137
```

18.5.1 Amarras na saída

Em algumas situações, você pode desejar que certos comandos sejam executados na saída de um script, mesmo se o usuário abordar a execução. Por exemplo, suponha que você tenha criado arquivos temporários, provavelmente desejará remover estes arquivo ao sair do script, mesmo que esta saída seja forçada pelo usuário com o envio de um sinal de interrupção (SIGINT, através da combinação Ctrl+C) ou termino (SIGTERM, através do comando **kill**) .

O exemplo abaixo ilustra como podemos remover os arquivos temporá-

rios criados mesmo quando o script é interrompido. Note que as amarras na saída só serão executadas se o processo não for abruptamente terminado por um sinal de extermínio (SIGKILL através do comando **kill -9**). Experimente rodar o script na Listagem 18.12 e enviar cada um dos sinais de interrupção listados anteriormente (SIGINT, SIGTERM e SIGKILL). Verifique que o arquivo temporário foi removido nos dois primeiros casos, mas não no terceiro.

Listagem 18.12: Incluindo amarras na saída

```
#!/bin/bash
echo "este␣sript␣possui␣o␣seguinte␣ID:␣$$"
tmpfile=$(mktemp)
trap 'rm "$tmpfile"' EXIT
echo "arquivo␣temporário␣criado:␣$tmpfile"
sleep 10m
seq 1 10 > $tmpfile
exit 0
```

É possível também criar amarras diferentes para sinais de interrupção diferentes. Veja na Listagem 18.13.

Listagem 18.13: Incluindo amarras para sinais de interrupção

```
trap "script␣interrompido!" SIGINT SIGTERM
```

Outro exemplo util de criar amarras para a saída do script é quando utilizamos algum arquivo de *lock*. Criamos um arquivo de *lock* quando queremos que apenas uma instância do script seja executada por vez, impedindo assim potenciais problemas que poderiam surgir da sobreposição de múltiplas instâncias de um mesmo script rodando. O exemplo da Listagem 18.14 ilustra uma estrutura básica para fazer isso.

Listagem 18.14: Incluindo amarras para aparar arquivo de lock

```
#!/bin/bash
LOCKFILE=/var/lock/myscript.lock
[ -f $LOCKFILE ] && exit 0
trap "{␣rm␣-f␣$LOCKFILE␣;␣exit␣255;␣}" EXIT
touch $LOCKFILE

...

exit 0
```

18.6 Lendo dados dos Usuários

É possível utilizar dados digitados pelo usuário durante a execução de um script. Para receber estes dados utilizamos o comando **read** como na Listagem 18.15.

Listagem 18.15: Lendo valores do teclado

```
#!/bin/bash

echo -n "Digite␣alguma␣coisa:␣"
read VALOR

echo "Você␣digitou:␣$VALOR"
```

O resultado obtido pode ser visto na Listagem 18.16.

Listagem 18.16: Resultado do Script

```
$ ./scrip04.sh
Digite alguma coisa: Oi mundo
Você digitou: Oi mundo
```

18.7 Estrutura de Seleção

É possível usar estruturas de seleção para criar fluxos de execução alternativos.

A estrutura de seleção mais comum é o **if**. Na Listagem 18.17 apresentados a definição de sua estrutura básica.

Listagem 18.17: Estrutura de seleção if else

```
if [ condição 1 ]
then
   comando 1
   comando 2
   ...
elif [ condição 2 ] # o mesmo que else if
then
   comando 3
   comando 4
else
   comando 5
   comando 6
   ...
fi
```

Não deixe de utilizar os espaços em brancos entre a condição e o colchetes, para evitar erro de sintaxe.

Podemos utilizar colchetes simples **[...]** ou duplos **[[...]]**. O primeiro é compatível com o padrão POSIX, enquanto o segundo é uma versão mais moderna que também permite utilizar expressões regulares assim como operadores no estilo C.

Listagem 18.18: Exemplo de utilização do if

```
$ cat a.sh
#!/bin/bash
if [ $1 -gt 1 ]
then
   echo "$1␣>␣1"
else
   echo "$1␣<=␣1"
fi
$ ./a.sh 0
0 <= 1
$ ./a.sh 1
1 <= 1
$ ./a.sh 2
2 > 1
$ ./a.sh 3
3 > 1
```

O script da Listagem 18.18 pode utilizar operador no estilo C, se utilizarmos colchetes duplos. Veja a Listagem 18.19, que terá o mesmo comportamento.

Listagem 18.19: Exemplo de utilização do if

```
#!/bin/bash
if [[ $1 > 1 ]]
then
   echo "$1␣>␣1"
else
   echo "$1␣<=␣1"
fi
```

A Listagem 18.20 apresenta o resultado do uso de uma estrutura de seleção.

Listagem 18.20: Estrutura de Seleção

```
#!/bin/bash
echo -n "Digite␣um␣número:␣"
```

```
read NUMERO

if [[ $NUMERO -gt 5 ]]
then
  echo "Número␣é␣maior␣que␣5."
else
  echo "Número␣é␣menor␣ou␣igual␣a␣5."
fi
```

A Tabela 18.3 apresenta o resumo das operações de comparação.

Tabela 18.3: Operadores de Comparação

significado	operador	operador estilo C
igual	-eq	==
diferente	-ne	!=
maior que	-gt	>
menor que	-lt	<
maior ou igual a	-ge	>=
menor ou igual a	-le	<=

18.8 Operador de teste de arquivos

A Tabela 18.4 apresenta os operadores de teste de arquivos. Podemos utilizar estes operadores para testar se um arquivo existe, se o arquivo é um diretório, se temos permissão para escrever no arquivo, etc.

Vamos criar um script para fazer o download de um arquivo apenas se o arquivo ainda não existir no nosso computador. Veja a Listagem 18.21 com este exemplo. Na primeira vez que executar o script, ele fará o download da obra completa de Shakespeare. Nas proximas vez que exercutá-lo, ele verificará que o arquivo está presente e não fará nada.

Listagem 18.21: Teste de arquivos

```
#!/bin/bash
URL="http://www.gutenberg.org/files/100/100-0.txt"
OUTFILE=${URL##*/}
if [ ! -e $OUTFILE ]
then
  wget -q $URL -O $OUTFILE
fi
```

Tabela 18.4: Operadores de teste de arquivos

significado	operador
arquivo existe	-e
é um arquivo regular (não é um diretório)	-f
tamanho não nulo	-s
é um diretório	-d
é um dispositivo de bloco	-b
é um dispositivo de caractere	-c
é um pip	-p
é um link simbólico	-L
é um socket	-S
é um dispositivo terminal	-t
possui permissão para leitura	-r
possui permissão para escrita	-w
possui permissão para execução	-x
possui um id de grupo	-g
possui um id de usuário	-u
o arquivo é do usuário corrente	-O
o arquivo possui mesmo id de grupo que o usuário corrente	-G

18.9 Loops

É possível criar loops utilizando scripts para execução de tarefas repetitivas. O comando echo é utilizado para imprimir um frase e quando desejamos imprimir o conteúdo de uma variável utilizamos $ antes do nome da variável. A Listagem 18.22 apresenta um script que irá executar 5 vezes o comando **echo**.

Listagem 18.22: Utilizando Loops

```
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Repetindo $i"
done
```

A Listagem 18.23 apresenta o resultado do script.

Listagem 18.23: Resultado do Script

```
$ ./scrip02.sh
Repetindo 1
Repetindo 2
Repetindo 3
```

```
Repetindo 4
Repetindo 5
```

Existem outras maneiras de se criar loops como na Listagem 18.24.

Listagem 18.24: Utilizando Loops

```
#!/bin/bash
for (( x=1; x<=10; x++ ))
do
   echo "Repetindo $x "
done
```

O resultado da execução é apresentado na Listagem 18.25.

Listagem 18.25: Utilizando Loops

```
$ ./scrip03.sh
Repetindo 1
Repetindo 2
Repetindo 3
Repetindo 4
Repetindo 5
Repetindo 6
Repetindo 7
Repetindo 8
Repetindo 9
Repetindo 10
```

18.10 Manipulação de Strings

Bash suporta várias formas de manipular strings. Veremos algumas nessa secção.

Podemos obter o tamanho de uma string de diferentes formas. Duas delas são ilustradas na Listagem 18.26.

Listagem 18.26: Comprimento de uma string

```
$ string="abcde12345"
$ echo ${#string}
10
$ echo `expr length $string`
10
```

Podemos buscar e obter o tamanho de um padrão em uma string conforme ilustrado na Listagem 18.27, onde estamos buscando por uma sequência de caracteres [a-z] minúsculos. Como podemos ver no exemplo, o padrão de busca casou com a sequência 'abcde' que possui comprimento 5.

Listagem 18.27: Comprimento de uma string

```
$ echo `expr match "$string" '[a-z]*'`
5
```

Para obter o índice de onde ocorre um determinado padrão na string, podemos fazer como ilustrado na Listagem 18.28.

Listagem 18.28: Índice de um padrão

```
$ echo `expr index "$string" 12`
6
```

Para extrair uma parte de uma string podemos fazer como ilustrado na Listagem 18.29. No primeiro exemplo iremos extrair a string a partir da posição 3 até o final. Já o segundo exemplo extrai a substring de comprimento 4 começando na posição de índice 3. O último exemplo também tem o mesmo resultado, porém a enumeração dos índices começa em 1 (um) em vez de 0 (zero).

Listagem 18.29: Extraindo parte de uma string

```
$ echo ${string:3}
de12345
$ echo ${string:3:4}
de12
$ echo `expr substr $string 4 4`
de12
```

Podemos também fazer a extração da primeira sequência na string que case com um determinado padrão de busca (dado por uma expressão regular). A Listagem 18.30 apresenta um exemplo de extração de um padrão.

Listagem 18.30: Extraindo um padrão de uma string

```
$ string=abcABC123abc
$ echo `expr match "$string" '\([a-z]*\)'`
abc
```

O exemplo da Listagem 18.31 ilustra como remover um padrão de dentro de uma string. Podemos remover o padrão mais curto que case com a expressão de busca (utilizando apenas um #) ou podemos remover o padrão mais longo que case com a expressão de busca (utilizando para tanto dois #).

Listagem 18.31: Removendo um padrão de uma string

```
$ string=abcABC123abc123
$ echo ${string#a*c}
```

```
ABC123abc123
$ echo ${string##a*c}
123
```

Quando utilizamos o #, como no exemplo da Listagem 18.31, a busca pelo padrão é feita a partir do início da string. Caso queira reverter o sentido de busca, deve-se utiliza o %. Da mesma forma, quando utilizamos apenas um %, será feita a substituição da substring mais curta e quando utilizamos dois %, será realizada a substituição da string mais longa. Veja os exemplos na Listagem 18.32.

Listagem 18.32: Removendo um padrão de uma string (sentido de busca reverso)

```
$ echo ${string%1*3}
abcABC123abc
$ echo ${string%%1*3}
abcABC
```

Utilizando os conceitos vistos acima, podemos escrever um script simples para converter todas as imagens do diretório corrente de PNG para JPEG. Veja o exemplo na Listagem 18.33.

Listagem 18.33: Convertendo todos arquivos PNG em JPEG

```
$ for file in $(ls *.png); do convert $file ${file%png
    }jpg; done
```

Na Listagem 18.34 mostramos como substituir uma substring dentro de uma string. No exemplo em questão, queremos substituir a sequência ‘abc’ pela sequência ‘xyz’. Mostramos exemplos onde substituímos a primeira ocorrência da sequência que foi buscada (utilizando apenas uma barra /), exemplo onde substituímos todas ocorrências da sequência buscada (utilizando para tando barra dupla //) e também substituição de uma ocorrência começando a busca do início (#) ou do final (%).

Listagem 18.34: Substituição de uma substring

```
$ string=abcABC123abc123
$ echo ${string/abc/xyz}
xyzABC123abc123
$ echo ${string//abc/xyz}
xyzABC123xyz123
$ echo ${string/#abc/xyz}
xyzABC123abc123
$ echo ${string/%abc/xyz}
abcABC123abc123
```

18.11 Expressões Aritméticas

Podemos fazer operações aritméticas com inteiros em um script, que pode ser muito útil. Vejamos alguns exemplos simples de como realizar tais operações.

A Listagem 18.35 apresenta um primeiro exemplo bem simples. Note que os espaços em branco no primeiro exemplo são necessários para que a expressão aritmética seja devidamente interpretada.

Listagem 18.35: Usando expressões aritméticas simples

```
$ z=10 && z=`expr $z + 5` && echo $z
15
$ z=10 && z=$(($z+3)) && echo $z
13
$ z=10 && z=$((z+1)) && echo $z
11
```

18.12 Redirecionamento I/O

Existem três arquivos que ficam abertos para fluxo de informação. São eles: **stdin** (entradas de dados, como aquela fornecida pelo teclado), stdout (saída usualmente impressa na tela do terminal), e stderr (saída de erro que também é impressa na tela). Podemos redirecionar estes arquivos e quaisquer outros, comandando o fluxo de informação. Cada um dos arquivos abertos recebe um descritor. Convencionalmente, os três mencionados anteriormente recebem os seguintes descritores: 0 (stdin), 1 (stdout), e 2 (stderr).

Os operadores usados para redirecionamento são $>$, $<$ e | (pipe). Veremos alguns exemplos simples de utilização.

Muitas vezes queremos que o resultado de um comando seja salvo em um arquivo em vez de ser impresso na tela. Para tanto, utilizamos o operador $>$. No exemplo da Listagem 18.36, vamos salvar a lista de arquivos resultante do comando **ls**.

Listagem 18.36: Redirecionando a saída para um arquivo

```
$ ls -l *.png > lista
$ cat lista
-rw-r--r-- 1 leoca leoca 44481 out 11 10:26 temp.png
```

Suponha você queria acrescer à lista anterior de outros arquivos. Para tanto, podemos utilizar o operador ». Ele irá redirecionar os dados de forma a juntar no final do arquivo já existente. Se o arquivo não existir, irá criar um novo. Veja o exemplo na Listagem 18.37.

Listagem 18.37: Redirecionando a saída para juntar em um arquivo já existente

```
$ ls -l *.jpg >> lista
$ cat lista
-rw-r--r-- 1 leoca leoca 44481 out 11 10:26 temp.png
-rw-r--r-- 1 leoca leoca 20401 out  9 13:34 lena.jpg
```

Note que o comportamento padrão é redirecionar a saída padrão (stdout) para o local de destino escolhido. Entretanto, podemos redirecionar também utilizando os descritores para explicitar o que queremos redirecionar. A Listagem 18.38 apresenta alguns casos.

Listagem 18.38: Utilizando os descritores para fazer o redirecionamento

```
1>arquivo # redireciona o stdout para o arquivo
2>arquivo # redireciona o stderr para o arquivo
&>arqiuvo # redireciona ambos para o arquivo
```

Note que em todos os casos ilustrados na Listagem 18.38, podemos utilizar também » se quisermos juntar mais dados ao invés de sobrescrever.

A Listagem 18.39 apresenta como redirecionar de um descritor para outro. Isto pode ser útil quando queremos que saída de erro também seja redirecionada para a saída padrão.

Listagem 18.39: Redirecionando de um decritor para outro

```
i>&j # redireciona o arquivo com descritor i para
   aquele com descritor j

2>&1  # redireciona stderr para stdout.

algum_comando >>arquivo 2>&1
# adiciona tanto a saída stdout quando a stderr em um
   arquivo
```

Um dos redirecionamentos mais utilizados é o pipe. Com ele podemos redirecionar a saída de um programa para a entrada de outro. O exemplo da Listagem 18.40 iremos redirecionar as saídas dos comandos para fazer um processamento contínuo dos dados. Queremos listar todas as palavras em todos arquivos texto. Vamos concatenar todos arquivos texto. A saída será redirecionada para um comando **tr** que irá apagar os caracteres diferentes de [a-zA-Z] e espaços. O resultado novamente é redirecionado, desta vez para um novo **tr** que irá substituir espaços por quebras de linha. For fim, vamos redirecionar o resultado para o **sort**, que irá ordenar as linhas e em seguida o **uniq** fornecerá apenas as ocorrências únicas. O resultado final é redirecionado para um arquivo chamado 'palavras'.

Listagem 18.40: Utilizando o pipe para listar todas as palavras

```
$ cat *.txt | tr -dc 'a-zA-Z[[:space:]]' | tr '[[:
   space:]]' '\n' | sort | uniq > palavras
```

Podemos também utilizar um arquivo para fornecer a entrada de dados, utilizando para tanto o <. Por exemplo, suponha que você queira ordenar as linhas de um arquivo, poderá fazer como ilustrado na Listagem 18.41.

Listagem 18.41: Redirecionado um arquivo para entrada de dados

```
$ sort < arquivo
$ sort < arquivo > resultado
```

18.13 Expressões Regulares

Expressões regulares são expressões compostas de caracteres e meta-caracteres que são utilizadas para buscar padrões. Os meta-caracteres são aqueles que possuem uma interpretação além do seu sentido literal.

Dependendo da função que os meta-caracteres possuem eles podem desempenhar papel de um conjunto de caracteres, papel de âncora (delimitando a posição em que buscamos um padrão) ou papel de modificador (que pode expandir ou estreitar o padrão de busca).

Tabela 18.5: Meta-caracteres nas expressões regulares e seus significados

significado	meta-caractere
qualquer número de repetição (inclusive zero)	*
repetição de um ou mais	+
opcional	?
especifica o número de ocorrências	m,n
início de linha	^
fim de linha	$
conjunto de caracteres	[...]
dígitos	
d	
letras	
w	
espaço em branco (inclusive tabulação e quebra de linha)	
s qualquer caractere	.
A-Za-z	[:alpha:]
0-9	[:digit:]
A-Za-z0-9	[:alnum:]
espaços e tabulação	[:space:]

Listagem 18.42: Um exemplo simples para buscar números de telefone

```
$ grep -o '\+\?[0-9 -]\{7,\}' usuarios.txt
+55 31 1234-5678
 11 1111-2222
```

Listagem 18.43: Um exemplo mais elaborada para buscar números de telefone

```
$ grep -o '\+\?\([0-9]\{1,3\}\s*\)\{1,2\}\s*[0-9
   -]\{8,\}' usuarios.txt
+55 31 1234-5678
11 1111-2222
```

Listagem 18.44: Um exemplo para buscar e-mails

```
$ grep -o '\([a-z0-9_\.-]\+\)@\([\da-z\.-]\+\)\.\([a-z
   \.]\{2,6\}\)' usuarios.txt
leolca@ufsj.edu.br
alessandrovivas@ufvjm.edu.br
```

18.14 Funções

As funções funcionam como sub-rotinas que implementam alguma funcionalidade. Na Listagem 18.45 podemos verificar a sintaxe de declaração de uma função. A utilização dos parênteses é opcional.

Listagem 18.45: Definindo uma função

```
nome_da_funcao () {
  comandos
  ...
}
```

Vamos começar definindo uma função simples e verificando o resultado de sua execussão. Veja a Listagem **??**.

Listagem 18.46: um exemplos simples de função

```
$ helloworld () { echo "ola"; echo "teste␣123"; }
$ helloworld
ola
teste 123
```

As funções podem receber parâmetros posicionais. Veja o exemplo da Listagem 18.47, onde criamos uma função para contar, com intervalo de 1

segundo, até um determinado número que será fornecido como parâmetro da função.

Listagem 18.47: Uma função para contar

```
$ contar () { for i in `seq 1 $1`; do echo $i; sleep 1
   s; done }
$ contar 3
1
2
3
```

18.15 Números Aleatórios

A variável de Shell **$RANDOM** fornece uma variável pseudo-aleatória de 15 bits, gerando números no intervalo de 0 a 32.767.

Listagem 18.48: Gerando números pseudo-aleatórios

```
$ for i in {1..5}; do echo $RANDOM; done
836
26118
14321
13371
15092
```

Note que, se você setar o valor inicial da variável, será gerada um sequência determinística. Veja na Listagem 18.49 como é gerada a mesma sequência de número nas duas utilizações, quando utilizamos a mesma semente.

Listagem 18.49: Utilizando a mesma semente

```
$ paste <(RANDOM=$$; for i in {1..5}; do echo $RANDOM;
    done) <(RANDOM=$$; for i in {1..5}; do echo
   $RANDOM; done)
23462   23462
29          29
1537    1537
12046   12046
32505   32505
```

A variável **$RANDOM** é inicializada no boot, mas você pode inicializá-la novamentem usando, por exemplo, o numero de épocas.

Listagem 18.50: Utilziando o número de épocas como semente

```
$ RANDOM=`date +%s`; for i in {1..5}; do echo $RANDOM;
    done
```

```
13787
24058
30609
31003
32537
```

Para gerar números aleatórios quando for necessário uma aleatoriedade mais rigorosa, podemos utilizar os dispositivos **/dev/random** e **/dev/urandom**. Eles utilizam ruídos do ambiente coletados pelo sistema operacional para formar uma piscina de entropia. A Listagem 18.51 mostra como podemos verificar quanta entropia temos disponível no sistema.

Listagem 18.51: Entropia disponível (em bits)

```
$ cat /proc/sys/kernel/random/entropy_avail
3765
```

À medida que lemos dados dos dispositivos **/dev/random** ou **/dev/urandom**, utilizamos bits de entropia da piscina. A diferença entre os dois dispositivos é que **/dev/urandom** fornece uma fonte ilimitada de números aleatórios, ou seja, mesmo quando acaba a fonte de entropia, continua a gerar dados aleatórios. Por outro lado, **/dev/random** só gera dados aleatório quando há entropia disponível. Desta forma, o **/dev/random** fornece uma forma de gerar dados mais rigorosamente aleatórios, porém limitado ao esgotamento de entropia armazenada na piscina. O exemplo da Listagem 18.52 ilustra como escrever um arquivo de 1 megabyte com dados aleatórios.

Listagem 18.52: Escrevento um arquivo aletório de 1 megabyte

```
$ head -c 1M < /dev/urandom > /tmp/randomfile
```

No exemplo da Listagem 18.53 geramos uma sequência de 5 números aleatórios com 10 dígitos.

Listagem 18.53: Escrevento um arquivo aletório de 1 megabyte

```
$ for i in {1..5}; do RNDNUM=$(cat /dev/urandom | tr -
   dc '0-9' | fold -w 10 | head -n 1); echo $RNDNUM;
   done
3058937502
4873955894
3377019198
5131152095
3222979158
```

Note que após executar os exemplos anterior, esvaziamos boa parte da piscina de entropia. Veja a Listagem 18.54.

Listagem 18.54: Entropia disponível

```
$ cat /proc/sys/kernel/random/entropy_avail
194
```

Capítulo 19

AWK

Sumário

19.1 Introdução

Awk é uma linguagem muito importante para processamento de texto e dados tabulares. É muito útil quando desejamos fazer processamento massivo de texto, isto é, arquivos com grande volume de dados. A funcionalidade básica do **awk** é buscar padrões e aplicar determinadas ações nos casos em que os padrões foram localizados. A linguagem **awk** se difere das outras linguagens popularmente conhecidas por ser uma linguagem orientada a dados (à descrição dos dados que serão processados e as ações que serão realizadas sobre os dados).

Neste capítulo veremos uma breve introdução ao **awk**. Para uma visão mais aprofundada e com diversos exemplos, veja o livro de autoria de Arnold Robbins [6].

A sintaxe básica de um programa **awk** é apresentada na Listagem 19.1.

Listagem 19.1: Sintaxe básica de um progama em awk

```
padrao {acao}
padrao {acao}
```

Para rodar um programa em **awk** podemos passar o código como uma string ou podemos salvar em um arquivo e usar a opção **-f**. A Listagem 19.2 ilustra as duas formas.

Listagem 19.2: Sintaxe básica para rodar um programa awk

```
$ awk 'programa' arquivo_de_dados
$ awk -f aquivo_do_programa arquivo_de_dados
```

Iremos mostrar alguns exemplos simples de utilização do **awk**. Para tanto, vamos definir o arquivo de dados conforme ilustrado na Listagem 19.3.

Listagem 19.3: Arquivos de dados que será utilizado nos exemplos seguintes

```
$ cat dados
Joao    300     1.2     S
Jose    200     0.9     N
Ana     100     2.3     N
Ana     80      2       N
Ricardo 10      3.5     S
Maria   150     0.8     N
Nadia   90      3.1     S
```

O programa mais simples que podemos fazer é o programa que simplesmente imprime os dados. Veja a Listagem 19.4.

Listagem 19.4: Programa que apenas imprime dos dados

```
$ awk '{print}' dados
Joao     300     1.2     S
Jose     200     0.9     N
Ana      100     2.3     N
Ana      80      2       N
Ricardo  10      3.5     S
Maria    150     0.8     N
Nadia    90      3.1     S
```

Vamos elaborar um pouco e vamos agora exigir que nosso programa imprima apenas as linhas onde encontrar um 'N' maiúsculo. Vejamos o programa e o resultado na Listagem 19.5.

Listagem 19.5: Programa para imprimir apenas as linhas com um determinado padrão

```
$ awk '/N/{print}' dados
Jose     200     0.9     N
Ana      100     2.3     N
Ana      80      2       N
Maria    150     0.8     N
Nadia    90      3.1     S
```

Note na Listagem 19.5 que a linha com 'Nadia' também foi impressa, afinal possui o caractere 'N'. Se quisermos imprimir apenas as linhas em que a quarta coluna possui 'N', devemos utilizar as variáveis embutidas na linguagem. Em **awk**, podemos referenciar cada um dos campos com **$N** onde N representa o número do campo. No exemplo, queremos analisar apenas se o campo 4 (quarta coluna) possui o caractere 'N', desta forma, devemos proceder como ilustrado na Listagem 19.6.

Listagem 19.6: Programa que apenas imprime dos dados

```
$ awk '$4~/N/{print}' dados
Jose     200     0.9     N
Ana      100     2.3     N
Ana      80      2       N
Maria    150     0.8     N
```

Para imprimir apenas a segunda coluna dos dados, devemos fazer como ilustrado na Listagem 19.7.

Listagem 19.7: Imprimindo apenas uma coluna

```
$ awk '{print $2}' dados
300
200
100
```

```
80
10
150
90
```

Muitas vezes queremos sumarizar os dados. Para ilustrar, vamos inserir uma linha ao final para imprimir a soma da coluna 2 e o produto da coluna 3. Veja a Listagem 19.8.

Listagem 19.8: Programa somar e multiplicar dados em uma coluna

```
$ awk 'BEGIN{OFS="\t"; PROD3=1} {SOMA2+=$2; PROD3*=$3;
    print} END{print "Total",SOMA2,PROD3}' dados
Joao    300     1.2     S
Jose    200     0.9     N
Ana     100     2.3     N
Ana     80      2       N
Ricardo 10      3.5     S
Maria   150     0.8     N
Nadia   90      3.1     S
Total   930     43.1222
```

O exemplo da Listagem 19.8 é bem mais completo. Note que introduzimos dois blocos: BEGIN e END. Estes blocos serão executados uma única vez, no início e no final do programa, respectivamente. No bloco BEGIN vamos setar o valor da variável embutida OFS (*output field separator*) para o espaço de tabulação, para que o resultado impresso ao final tenha o mesmo padrão de espaçamento utilizado no arquivo de dados. No bloco BEGIN ainda inicializamos a variável PROD3 como sendo igual a um. Note que não é necessário inicializar a variável SOMA2 pois o padrão da linguagem é inicializar as variáveis com zero (ou vazio). Seguindo o código, temos o bloco principal que será executado em todas as linhas dos arquivo de dados (uma vez que não especificamos uma condição para o bloco ser executado) em que fazemos a soma da segunda coluna o produto da terceira e imprimimos a linha. Por fim, o bloco END irá imprimir o resultado da soma e produto.

Usando agora a ideia vista na Listagem 19.6 e a vista na Listagem 19.8, podemos facilmente fazer um exemplo onde iremos sumarizar os dados de acordo com a última coluna. Iremos somar e multiplicar separadamente de acordo com o valor da última coluna. Poderíamos ter criado outras variáveis para sumarizar os dados, ou utilizar um array, como veremos na Listagem 19.9.

Listagem 19.9: Programa que apenas imprime dos dados

```
$ awk 'BEGIN{OFS="\t"; PROD3["N"]=1; PROD3["S"]=1} $4~
   /N/{SOMA2["N"]+=$2; PROD3["N"]*=$3} $4~/S/{SOMA2["S
   "]+=$2; PROD3["S"]*=$3} {print} END{print "Total_N"
```

```
   ,SOMA2["N"],PROD3["N"]; print "Total_S",SOMA2["S"],
   PROD3["S"]}' dados
Joao     300     1.2     S
Jose     200     0.9     N
Ana      100     2.3     N
Ana      80      2       N
Ricardo  10      3.5     S
Maria    150     0.8     N
Nadia    90      3.1     S
Total_N  530     3.312
Total_S  400     13.02
```

19.2 Controle de Fluxo

De forma similar à maioria das linguagens de programação, o **awk** também possui controle de fluxo. Na Listagem 19.10 apresenta a sintaxe básica para ser utilizada.

Listagem 19.10: Controle de fluxo If Else

```
if (conditicao)
   acao

if (conditicao) {
   acao1
   acao2
}

if (condicao)
   acao1
else
   acao2
```

A versão GNU do **awk**, chamada **gawk**, possui ainda o *switch*. Veja a sintaxe básica na Listagem 19.11.

Listagem 19.11: Controle de fluxo switch

```
switch (expressao) {
case valor_ou_expr_reg:
    acao1
default:
    acao2
}
```

Suponha que queremos imprimir as linhas do arquivo de dados, visto na Listagem 19.3, e quando a soma dos valores na segunda coluna ultrapassar um determinado limiar (no exemplo utilizamos o valor 310), queremos imprimir o valor da soma parcial. Veja o programa ilustrado na Listagem 19.12.

Listagem 19.12: Controle de fluxo switch

```
$ awk '{print; SOMA+=$2; if (SOMA>310) {print "PARCIAL
   ␣=",SOMA; SOMA=0;}}' dados
Joao     300     1.2     S
Jose     200     0.9     N
PARCIAL = 500
Ana      100     2.3     N
Ana      80      2       N
Ricardo  10      3.5     S
Maria    150     0.8     N
PARCIAL = 340
Nadia    90      3.1     S
```

19.3 Loops

A linguagem **awk** também possui estrutura de loop utilizando **for**, **while** e **do-while**.

Os exemplos apresentados na Listagem 19.13 mostram uma simples utilização das estruturas de loop para imprimir os números de 1 a 5.

Listagem 19.13: Exemplo simples de loop em awk

```
$ awk 'BEGIN {for (i = 1; i <= 5; ++i) print i}'
$ awk 'BEGIN {while (i < 5) print ++i }'
$ awk 'BEGIN {do {print ++i} while (i < 5) }'
```

19.4 Funções

Na linguagem **awk** também podemos definir funções. A estrutura básica para definir uma função é apresentada na Listagem 19.14.

Listagem 19.14: Estrutura de funções em awk

```
function nome_da_funcao(argumento1, argumento2, ...) {
  acao1
  acao2
}
```

Iremos utilizar exemplos anteriores e vamos definir uma função para retornar o mínimo de dois números. Podemos utilizar esta função em todas as linhas para buscar qual é o valor mínimo da segunda coluna. Iremos necessitar de uma variável para guardar o valor do mínimo corrente. Veja o exemplo na Listagem 19.15.

Listagem 19.15: Exemplo de utilização de função

```
$ awk 'function min(a,b){if(a<=b) return a; else
   return b;} BEGIN{omin=9999} {print; omin=min(omin,
   $2)} END{print "minimo = "omin}' dados
Joao    300     1.2     S
Jose    200     0.9     N
Ana     100     2.3     N
Ana     80      2       N
Ricardo 10      3.5     S
Maria   150     0.8     N
Nadia   90      3.1     S
minimo = 10
```

19.5 Redirecionamento

Dentro de um programa em **awk** podemos fazer o redirecionamento da saída, da mesma forma que fazer no **shell**.

Vejamos na Listagem 19.16, um exemplo de como redirecionar a saída para um arquivo.

Listagem 19.16: Redirecionando a saída para um arquivo

```
 awk 'BEGIN { print "Hello World" > "/tmp/test.txt" }'
$ cat /tmp/test.txt
Hello World
```

Também podemos redirecionar a saída para outro comando utilizando pipe, assim como fazemos em um script bash. Veja o exemplo na Listagem 19.17.

Listagem 19.17: Redirecionando a saída para um outro comando utilizando pipe

```
$ awk 'BEGIN { print "Hello World" | "tr [a-z] [A-Z]"
   }'
HELLO WORLD
```

Podemos também fazer uma comunicação em ambos sentidos com outros comandos. Para tando devemos utilizar o operador |&. O exemplo na Lis-

tagem 19.18 ilustra como fazer uma comunicação em duplo-sentido. Nesse exemplo iremos ordenar os caracteres de cada linha.

Listagem 19.18: Comunicação em ambos sentidos

```
$ cat sort.awk
BEGIN {
   cmd = "grep␣-o␣.␣|␣sort␣|␣tr␣-d␣'\n'"
}
{
   print |& cmd
   close(cmd, "to")

   cmd |& getline out
   print out;
   close(cmd);
}
$ echo -e 'hello world!\nhow are you?' | awk -f sort.
   awk
 !dehllloorw
  ?aehooruwy
```

Índice

Bibliografia

[1] Canonical Ltda. Ubuntu, 2019.

[2] Debian. Debian, 2019.

[3] Fedora. Fedora, 2019.

[4] Linuxize. Linux Time Command, 2019.

[5] OpenSuse. OpenSuse, 2019.

[6] Arnold Robbins. *Effective awk Programming (3rd Edition)*. O'Reilly Media, 2001.

[7] Ellen Siever, Stephen Figgins, Robert Love, and Arnold Robbins. *Linux in a Nutshell: A Desktop Quick Reference*. O'Reilly Media, 2009.

[8] A. Silberschatz, P.B. Galvin, and G. Gagne. *Operating System Concepts, 9th Edition*. Wiley, 2012.

[9] A.S. Tanenbaum and H. Bos. *Modern Operating Systems*. Always learning. Pearson, 2015.

[10] VirtualBox. Virtual Box, 2019.

[11] Wikibooks. Sistemas operacionais/gerência de dispositivos de entrada e saída. `https://pt.wikibooks.org/wiki/Sistemas_operacionais/Gerência_de_dispositivos_de_entrada_e_saída`, 2019. [Online; accessed 28-August-2019].

[12] Wikipedia. AMD64 — Wikipedia, the free encyclopedia. `http://pt.wikipedia.org/w/index.php?title=AMD64&oldid=56090401`, 2019. [Online; accessed 28-August-2019].

[13] Wikipedia. Distribuição Linux — Wikipedia, the free encyclopedia. `http://pt.wikipedia.org/w/index.php?title=Distribui%C3%A7%C3%A3o%20Linux&oldid=49962639`, 2019. [Online; accessed 28-August-2019].

[14] Wikipedia. Peripheral Component Interconnect — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Peripheral%20Component%20Interconnect&oldid=55719137, 2019. [Online; accessed 28-August-2019].

[15] Wikipedia. Unidade central de processamento — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Unidade%20central%20de%20processamento&oldid=56064143, 2019. [Online; accessed 28-August-2019].